CONSISTENT FEATURE EXTRACTION FROM VECTOR FIELDS: COMBINATORIAL REPRESENTATIONS AND ANALYSIS UNDER LOCAL REFERENCE FRAMES

by

Harsh Bhatia

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

May 2015

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Harsh Bhatia**

has been approved by the following supervisory committee members:

Valerio Pascucci	, Chair	**11/24/2014** Date Approved
Adam W. Bargteil	, Member	**10/30/2014** Date Approved
Peer-Timo Bremer	, Member	**10/30/2014** Date Approved
Christopher R. Johnson	, Member	**10/30/2014** Date Approved
Robert M. Kirby	, Member	**10/30/2014** Date Approved

and by **Ross Whitaker**, Chair/Dean of

the Department/College/School of **School of Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

With modern computational resources rapidly advancing towards exascale, large-scale simulations useful for understanding natural and man-made phenomena are becoming increasingly accessible. As a result, the size and complexity of data representing such phenomena are also increasing, making the role of data analysis to propel science even more integral. This dissertation presents research on addressing some of the contemporary challenges in the analysis of *vector fields*—an important type of scientific data useful for representing a multitude of physical phenomena, such as wind flow and ocean currents. In particular, new theories and computational frameworks to enable ***consistent feature extraction from vector fields*** are presented.

One of the most fundamental challenges in the analysis of vector fields is that their features are defined with respect to reference frames. Unfortunately, there is no single "correct" reference frame for analysis, and an unsuitable frame may cause features of interest to remain undetected, thus creating serious physical consequences. This work develops new reference frames that enable extraction of localized features that other techniques and frames fail to detect. As a result, these reference frames objectify the notion of "correctness" of features for certain goals by revealing the phenomena of importance from the underlying data. An important consequence of using these local frames is that the analysis of unsteady (time-varying) vector fields can be reduced to the analysis of sequences of steady (time-independent) vector fields, which can be performed using simpler and scalable techniques that allow better data management by accessing the data on a per-time-step basis.

Nevertheless, the state-of-the-art analysis of steady vector fields is not robust, as most techniques are numerical in nature. The residing numerical errors can violate consistency with the underlying theory by breaching important fundamental laws, which may lead to serious physical consequences. This dissertation considers consistency as the most fundamental characteristic of computational analysis that must always be preserved, and presents a new discrete theory that uses combinatorial representations and algorithms to provide consistency guarantees during vector field analysis along with the uncertainty visualization of unavoidable discretization errors.

Together, the two main contributions of this dissertation address two important concerns regarding feature extraction from scientific data: correctness and precision. The work presented here also opens new avenues for further research by exploring more-general reference frames and more-sophisticated domain discretizations.

To my parents, Ashok and Kusum Bhatia,
for their love and support, and
for instilling in me the strength and self-belief to pursue my dreams!

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

nD	n-dimensional
$\mathbb{R}^n$	nD Euclidean space
$\mathbb{B}^n$	nD Unit ball
$\mathbb{M}$	Manifold
Ω	Bounded subset of $\mathbb{R}^n$
$\partial\Omega$, $\mathring{\Omega}$, and $\bar{\Omega}$	Boundary, Interior, and Closure of Ω
$\mathcal{S}^k$	k-simplex
Δ	Triangle ($\mathcal{S}^2$)
$\mathcal{V}$, $\mathcal{E}$, and $\mathcal{T}$	Sets of vertices, edges, and triangles
$\mathcal{M} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}\}$	Simplicial mesh
$\mathcal{G}$	Regular grid
V and E	Sets of vertices and edges of a graph
$\mathsf{G} = \{\mathsf{V}, \mathsf{E}\}$	Graph
$\{x, y, z\}$	Euclidean coordinates
$\{\rho, \theta, \phi\}$	Spherical coordinates
$\{i, j, k\}$	Indices
t	Time
$\mathbf{x}, \mathbf{p}, \mathbf{q}, \ldots$	Points in $\mathbb{R}^n$
$\mathbf{X}, \mathbf{P}, \mathbf{Q}, \ldots$	Sets of contiguous points in $\mathbb{R}^n$
$f, g, \ldots$	Scalar fields
$\vec{V}(\mathbf{x})$	Time-independent (steady) vector field
$\vec{V}(t, \mathbf{x})$	Time-varying (unsteady) vector field
$\mathrm{St}(\mathbf{v})$	Star of a vertex $\mathbf{v}$
$\mathfrak{J}$ and $\mathfrak{H}$	Jacobian and Hessian
ε and δ	Spatial and temporal errors
$\{\ldots\}$	Set

PREFACE

This dissertation is submitted for the degree of Doctor of Philosophy at the University of Utah, Salt Lake City. The research described herein was conducted by the author under the supervision of Prof. Valerio Pascucci at the Scientific Computing and Imaging (SCI) Institute, the University of Utah, and Dr. Peer-Timo Bremer at the Center for Applied Scientific Computing, Lawrence Livermore National Laboratory (LLNL). This dissertation describes original work by the author, including notable contributions of Shreeraj Jadhav, M.S., in Chapters 6 and 7, and except where acknowledgments and references have been made to previous work.

In complete accord with Sir Isaac Newton's statement, *"If I have seen farther, it is by standing on the shoulders of giants!"*, I would like to acknowledge the people who made this dissertation possible by helping me through graduate school: personally or professionally; technically or philosophically.

I would like to thank, first of all, my parents, who taught me to ask questions and to seek answers, and instilled in me the strength and self-belief to pursue my dreams; and my sister, who always supported me and understood me for all my personal and professional decisions. I would not be where I am today without the love, support, and encouragement from my family across time zones, and their patient understanding when I would just refuse to talk during the long debugging hours with my code!

I have been extremely fortunate to find a great advisor in Prof. Valerio Pascucci. As an academic mentor, Valerio not only provided technical guidance, but also helped me gain a broader insight into the challenges, scope, and future impact of this research. Even more importantly, he also listened to my professional and personal conundrums, and helped me steer out. His work ethics, attention to detail, and philosophical perspectives have been a continuous source of motivation, for which I look up to him with admiration.

The last two and a half years of my Ph.D. were spent at LLNL working closely with my mentor, Dr. Peer-Timo Bremer. Working with Timo has been a pleasure; being able to just walk across the hall and have detailed discussions—technical or nontechnical—has been immensely useful. Special thanks go to Timo for his detailed feedback on my technical

writing, and continuously helping me grow. This dissertation would not be complete without his knowledge, experience, and support.

I would also like to thank my other committee members: Prof. Chris Johnson, Prof. Mike Kirby, and Dr. Adam Bargteil, who in the midst of their busy schedules found time to guide me and provide feedback on my research and progress. Dr. Joshua Levine played an important role in converting a nonspecialist fresh graduate student into the knowledgeable visualization Ph.D. I am today; I am immensely grateful to him for his guidance. I would like to thank all my collaborators for their valuable contributions to this work. I am also thankful to my numerous colleagues at the SCI Institute and LLNL, who have inspired me by their sheer brilliance.

Last but not the least, I would like to thank all my friends around the globe for their love and support, and providing the much-needed distractions from long research hours.

The research presented in this dissertation was funded in part by the LLNL Graduate Scholars Program, and is not a deliverable for any United States government agency. The views and opinions expressed are those of the author, and do not state or reflect those of the United States government or Lawrence Livermore National Security, LLC. A part of this work was performed under the auspices of the US Department of Energy (DOE) by Lawrence Livermore National Laboratory (LLNL) under contract DE-AC52-07NA27344. LLNL-TH-663976.

Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Harsh Bhatia

January 31, 2015

CHAPTER 1

INTRODUCTION

"Without understanding data,
you're just another person with an opinion!"

Data analysis is an expansive field consequential for multifaceted scientific progress. Typically, all of our scientific endeavors to understand natural and artificial phenomena generate data containing a wealth of information. In order to uncover the mysteries of the universe, it is imperative to analyze this data. The typical role of data analysis in the process of scientific discovery is illustrated in Figure 1.1.

The contemporary notion of data analysis may appear pertinent only to modern and complex scientific data; nevertheless, the earliest and much simpler ideas of analyzing data can be traced back to ancient scientific history. According to a famous anecdote, about 2200 years ago, Archimedes, the great Greek mathematician and philosopher, was posed with the problem of investigating whether a golden crown contained impurities in the form of silver—a lesser valued metal. Using the densities of the two metals, known at his time, Archimedes came up with the ingenious idea of analyzing the volume of water displaced by the crown in order to accurately calculate the amounts of gold and silver present in the

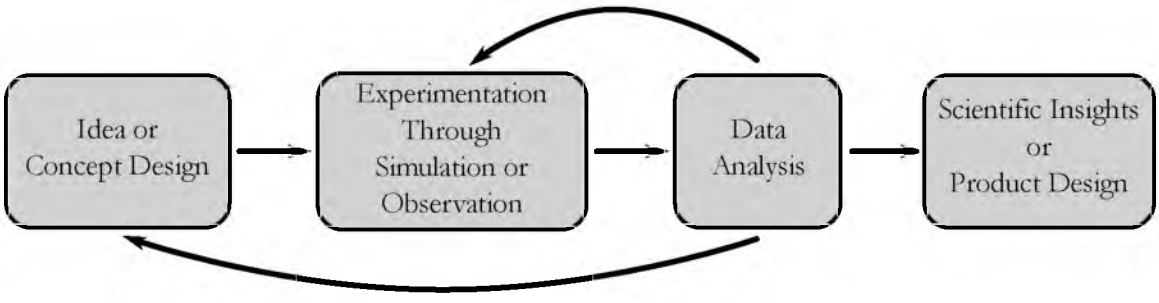


Figure 1.1. Data analysis plays an integral role in the process of scientific discovery, as it allows drawing important insights from the underlying phenomena, and provides feedback enabling improvement of hypotheses and designs.

crown—a concept used to date, and known as the Law of Buoyancy.

More than two millennia later, while science has made enormous progress, our quest for understanding the unknown still exists. Pioneered by the early inventions of Blaise Pascal and Charles Babbage, the use of calculators and computers has increased immensely in the past century. This increasing use has allowed relegating many tedious tasks to machines, thus freeing the human mind to seek the answers to questions with much greater significance and complexity. Modern scientific progress relies greatly on the strong collaborations between domain scientists and computer scientists, where domain scientists perform experiments to understand important natural or artificial physical processes, and the role of computer scientists is to enable new technologies to simulate and/or analyze these phenomena. At the time of this dissertation, petascale[1] computing is already available, and efforts are being made to achieve exascale[1] capabilities. Such high-performance computing (HPC) resources equip the scientists for performing simulations at previously impossible proportions, generating large-scale complex data to represent underlying physical phenomena ranging from subatomic to astronomical.

1.1 Types of Scientific Data

Data resulting from scientific experiments can have many forms, each with different characteristic features. For example, data can be spatial or nonspatial, continuous or discrete, single- or multivalued, *steady* (time-independent) or *unsteady* (time-dependent), etc. Common types of nonspatial data include text- or graph-based data. Spatial data is typically defined on one-dimensional (1D), two-dimensional (2D), or three-dimensional (3D) Euclidean spaces, or surfaces (sometimes referred to as 2.5D), with a possible additional dimension of time. Most common forms of spatial data found in scientific experiments are steady or unsteady scalar, vector, and tensor fields.

A *scalar field* represents the magnitude of a property at each spatial location, for example, temperature and pressure for a geographical region. Scalar fields have been studied extensively in the past few decades, with a large number of efficient techniques for analyzing them. *Vector fields* generalize scalar fields as they represent properties with both a direction and a magnitude. Common examples of vector fields are velocity fields, fluid flows, magnetic fields, and gradients of scalar fields. *Tensor fields* further generalize vector fields, and can

[1]Petascale and exascale computing, respectively, can perform 10^{15} and 10^{18} floating-point operations per second.

be imagined as assigning a vector of variable length and direction to each point in space, for example, the gradient of vector fields, and stress and strain tensors.

Different types of data have different characteristic features, and require different types of analysis. From scalar to vector to tensor data, the complexity increases and so do the challenges in the corresponding analysis. Furthermore, regardless of the type of data, certain challenges arise due to the computational nature of analysis frameworks.

1.2 Computational Challenges in Data Analysis

The analysis of the data—in particular, large-scale complex data—poses several challenges to computer scientists. Contemporary computational challenges are difficult, and addressing even a single one of them in depth may require one or more dedicated Ph.D. dissertations. Although the primary goal of this dissertation is not to provide comprehensive solutions to these computational challenges in general, it is important to develop a good understanding of them, as the research presented in this dissertation develops new and effective analysis techniques that address some of these computational challenges in the context of vector fields.

1.2.1 Challenges Due to the Numerical Nature of Analysis

Although computing resources can work at extremely high speeds, the underlying processors are binary in nature. They must convert smooth analog signals to nonsmooth digital representations, which limits their ability to represent signals of arbitrary precision. A direct consequence of this limitation is that real numbers, which have infinite precision, must be represented using floating-point numbers, which have finite precision. Numerical representation of data using floating-point numbers on computing machines, and the use of floating-point arithmetic to analyze such data, create unavoidable errors [70]. Since computational models for data analysis are adapted from mathematical models defined for smooth functions, it is widely accepted that any computer application based on numerical techniques can produce only an approximation to theoretical results, unless the theory itself contains a discrete model.

Nevertheless, the fundamental numerical nature of computing can cause serious problems, especially when the same application is run on different machines with different underlying architectures or using different compilers. Although numerical errors cannot be eliminated altogether, any application must be able to at least guarantee obtaining the same, albeit approximate, results on different platforms. Therefore, achieving numerical robustness is an important characteristic for analysis applications. Understanding numerical

robustness is one of the main goals of scientific computing and numerical analysis, and has led to various research areas such as numerical verification and validation [152], sensitivity and uncertainty analysis [21], etc. In particular, we focus on two kinds of problems that arise due to the lack of numerical robustness in computational models.

1.2.1.1 Inaccuracy and uncertainty. This work uses the term *inaccuracy* to refer to the varying degrees of accuracy in analysis. It can mean obtaining an approximate solution instead of an exact one, or can imply the existence of false positives or false negatives in the results. For example, through the analysis of a projectile motion, one can determine the value of gravitational acceleration to be 10 m/s^2, 9.8 m/s^2, or 9.81 m/s^2, reflecting multiple levels of inaccuracy. Another example of inaccuracy is the incorrect identification of cancerous cells in the analysis of a brain's magnetic resonance imaging (MRI) scan. In general, inaccuracy is closely related to the notion of *uncertainty*, where a certain confidence level is associated with the results, for example, considering the value of gravitational acceleration to be accurate up to 0, 1, or 2 decimal digits, or considering it to be only 80% likely that a certain region shown in the MRI scan contains cancerous cells. As a result, uncertainty quantification and uncertainty visualization are integral to the process of analysis for deriving scientific insights.

One way of addressing the problem of inaccuracy can be to represent data and perform numerical arithmetic with more precision, for example, using 64 bits for representation and processing of floating-point data instead of 32 bits. Although this approach may be suitable for certain applications, it is not sufficient for data analysis for two reasons. First, the precision of the representation cannot be increased arbitrarily, and the precision required to faithfully analyze a given data may not be known in advance. The second and more important reason is that the goal of data analysis is to understand some physical process when its true behavior is not known. In the absence of the "ground truth", it is not possible to define inaccuracy in an objective manner. For example, during the study of the MRI scan, analysts do not know where the actual cancerous cells are. In fact, identification of the ground truth in this case requires cutting through the patient's brain—a task with huge physiological costs that defeats the purpose of exploratory analysis altogether.

1.2.1.2 Inconsistency. In the example of projectile motion discussed above, depending upon the error tolerance of the application, various levels of inaccuracy may be accepted. However, if at any instant, the analysis shows that the gravitational acceleration is acting upward instead of downward, then the analysis immediately loses all merit, because this result is inconsistent with the fundamental theory of gravity. This dissertation uses

the term *inconsistency* to represent scenarios where the analysis produces physically invalid results such as the one discussed above. Restated, inconsistency implies that the results of computational analysis are not consistent with the underlying theory, and certain rules and invariants of the theory are violated.

It should be noted that inconsistency is not an extreme case of inaccuracy, but can be highly data-dependent, and can be caused by only small amounts of errors. Section 1.4.4 will discuss inconsistency in the context of vector fields with specific examples to motivate the need for consistent analysis, demonstrating that inconsistent analysis produces physically meaningless results, and any practical insights drawn from such analysis are incorrect and can lead to disastrous consequences. These examples insist that inconsistent analysis is far worse than inaccurate analysis.

The work presented in this dissertation is founded upon the notion that consistency is the most fundamental characteristic of analysis. Unlike inaccuracy, which requires knowing the true results, inconsistency can be defined objectively without requiring the knowledge of the ground truth. Whereas inaccuracy is simply a function of the precision of the application, we consider consistency to be more important and an objective metric for numerical robustness, and argue that a consistent but potentially less accurate analysis may be favorable over a more accurate yet potentially inconsistent solution.

1.2.2 Challenges Due to the Large Scales of Data

Most forms of innovation start with small and simple ideas and experiments. With the evolution of ideas, however, it is crucial to scale them up to allow a real scientific impact. Modern HPC resources allow generation of terabytes and petabytes of complex data that need to be analyzed. At the same time, these computational resources also offer opportunities to design *scalable* and *parallel* applications for analysis. A typical supercomputer contains thousands of computing nodes with each node further containing multiple processing units. In order to utilize these resources, algorithms and applications must be designed in a scalable and parallelizable manner, so as to run simultaneously on the many cores of supercomputers. The task of parallelization is often nontrivial and has many constituents, for example, designing parallel file systems, optimizing internode communication, synchronization, load-balancing, etc. As a result, scalability and parallelization of applications are important computational challenges that must be addressed.

As modern large-scale simulations and experiments continue to generate increasing amounts of data, efficient *storage* and *management* of the resulting data also becomes important. The data-access patterns of computer applications are of particular importance, since

loading blocks of data—especially noncontiguous blocks—in and out of memory typically creates performance bottlenecks. Therefore, applications that require frequent reloading of blocks of data into memory are inefficient and potentially prohibitive for large-scale data. Furthermore, since the compute power of modern machines continues to grow faster than the corresponding input/output (I/O) rates, the latter disallow storing the amount of data required for faithful analysis. For example, although highly accurate simulations are being used to generate data, due to I/O bottlenecks, typically data for only the 500^{th} simulation time-step is stored on disk. This reduced temporal resolution is often suboptimal to capture the behavior of the underlying process in a faithful manner. One solution to address this limitation is to develop in situ techniques, where the simulation or the experiment is temporarily halted, the current state of data is analyzed, and only the analysis results are written to disk. However, such in situ analysis techniques must be designed to perform a temporally local analysis, since the information corresponding to the past and the future is usually not available.

1.3 Vector Fields and Their Analysis

Vector fields assign a direction and a magnitude to every point in the domain. They are an important form of data, and are used to represent a wide variety of natural and artificial phenomena. For example, vector fields represent wind patterns [15] and ocean currents [133] useful for the modeling and analysis of weather and aerodynamical designs. Vector fields are used to capture the flow behavior of fuel in the study of turbulent combustion to understand mixing, circulation, and transport problems [88, 222, 223]. Astronomical phenomena such as solar convection in astrophysics [194] are also represented using vector fields. Vector fields also represent force fields such as gravitational and magnetic fields. Dynamical systems in nature, such as the balance between the populations of prey and predator species, can also be represented as vector fields. Some of the many use cases of vector fields are shown in Figure 1.2. Due to the wide applicability of vector fields, their analysis is indispensable in a large number of applications.

Most vector fields of interest change with time, and are called *unsteady* or time-varying vector fields, whereas those vector fields that do not change with time are called *steady* vector fields. In most common scenarios, vector fields represent flow of material, such as ocean currents and wind patterns; therefore, they are often simply called *flows*.

Similar to the analysis of other forms of data, a typical goal of vector field analysis is to extract important features, and understand the overall behavior of the data in a qualitative

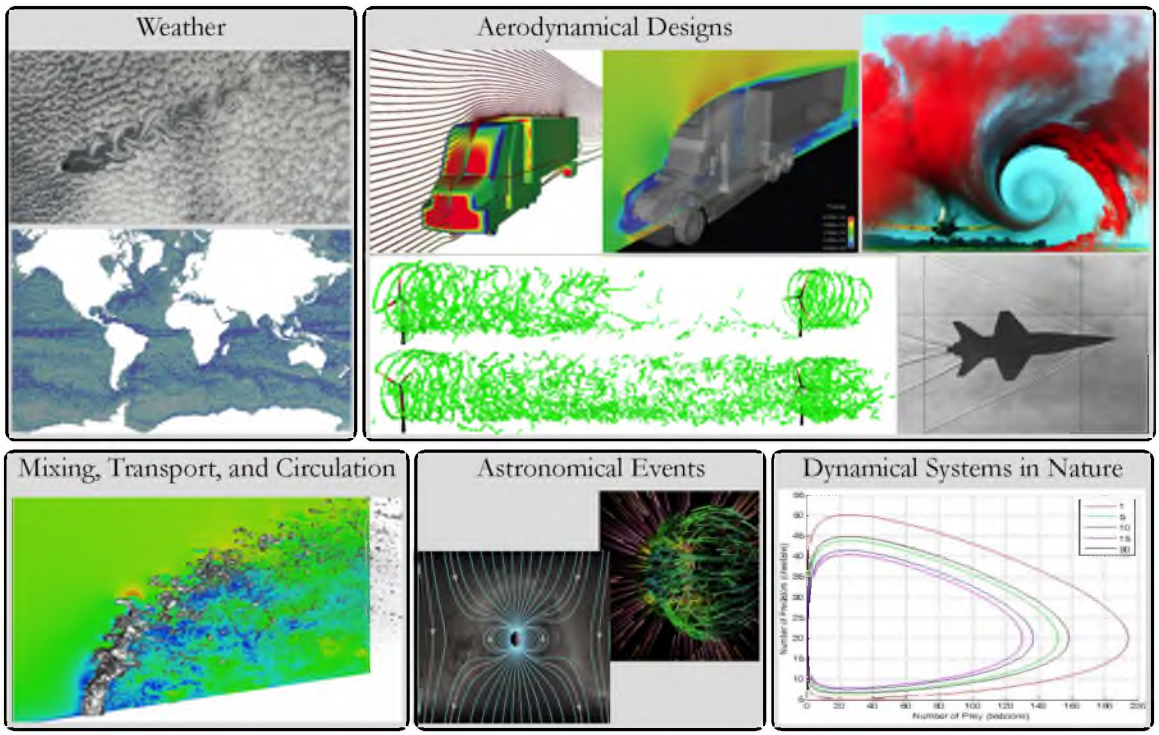


Figure 1.2. Vector fields are used to represent a wide variety of physical phenomena.

and/or a quantitative manner. In the case of vector fields, features of interest include *vortices* (regions of rotating material), *critical points* (points where material is created or destroyed), *material boundaries* (boundaries that the material cannot cross), *separation and attachment lines* (where flow separates and attaches, respectively), *shocks* (discontinuities in the field), etc. The techniques for vector field analysis often aim to extract some or all types of features.

1.3.1 Paradigms of Vector Field Analysis

The techniques for vector field analysis can be broadly classified into three classes briefly discussed in this section. A more detailed discussion along with the advantages and limitations of these classes of techniques can be found in Chapter 2.

1.3.1.1 Pathline-based analysis. *Pathlines* are paths of massless[2] material particles as they are advected in the vector field. Since particle paths capture the overall dynamic behavior of the material represented by the vector field (wind, ocean, etc.), this class forms the most fundamental type of analysis of vector fields. A large number of analysis techniques

[2]Particles are assumed to be massless so that they do not have any inertia, and their motion represents the influence of the underlying vector field only.

compute these paths—through numerical integration—and use them either directly [203] or indirectly [80, 81] to extract features of interest.

1.3.1.2 Streamline-based analysis. *Streamlines* are curves that are instantaneously tangent to the vector field at every point in space. They encode the direction that a moving particle will take at a given point in space and time, and, therefore, describe the spatial behavior of the field at a given instant in time. In the case of steady vector fields, streamlines capture the particle behavior, and, therefore, coincide with pathlines. Consequently, streamline-based analysis and visualization [20, 89, 208] have been very successful for steady vector fields.

1.3.1.3 Indicator-based analysis. The third class of vector field analysis techniques extracts the features of interest indirectly through carefully chosen *scalar indicator functions* that may represent position-based or particle-based properties of the data. For example, vortices can be extracted using indicators such as Q- [97] and λ_2-criterion [100]. The primary advantage of using scalar indicators is that it allows utilizing robust scalar analysis techniques instead of more difficult vector field analysis.

1.4 Challenges in Vector Field Analysis

Due to the fundamental dynamic nature of vector fields, their analysis is particularly challenging—especially as compared to the analysis of scalar fields. This section discusses some important mathematical and algorithmic challenges in vector field analysis, many of which will be addressed by this dissertation.

1.4.1 Challenges Due to Nontrivial Features

Perhaps the most fundamental challenges in vector field analysis exist in the understanding of complex features. Although 2D steady vector fields have been studied thoroughly in the past two decades using effective techniques such as vector field topology [89], the more important cases—the 3D and the unsteady—are still not well understood, especially because features in such vector fields are more complex and often nontrivial to define. In particular, many important features are defined only intuitively, and a rigorous mathematical formulation does not exist.

For example, vortices are important to identify in a variety of physical applications. They induce drag and can reduce lift in airplanes; hence, they must be minimized. On the other hand, in combustion and mixing phenomena vortices may be desirable as they can increase stirring. Nevertheless, despite their importance, vortices do not have a universally accepted definition—a challenge well represented by the numerous research articles in the

field that try to define vortices, for example, *"The Dilemma of Defining a Vortex"* [97] and *"The Objective Definition of a Vortex"* [82]. Some of the most common techniques to detect vortex derive indicators such as Q- [97] and λ_2-criterion [100], which highlight vortical properties of the flow that may suffice as a proxy for vortices. Similarly, material boundaries (boundaries that separate material) in unsteady flows are also not well understood. They are often extracted indirectly as the ridges of the finite-time Lyapunov exponent [80, 81], which has also been questioned [83].

1.4.2 Challenges Due to the Dependence Upon Reference Frames

Fields are physical quantities that are represented—both mathematically and computationally—as functions of space and time. Such a representation must assume a *reference frame* to define a coordinate system. It is well known from classical relativity that the velocity of an object as measured by a moving observer (reference frame) depends upon the observer's velocity itself [112]. Therefore, vector fields, which define velocity fields, vary with moving reference frames. They must be measured and analyzed in an implicitly assumed reference frame; streamlines and streamline-based features such as critical points, orbits, etc., can be defined only for a given reference frame. With a changing reference frame, such features may move around, appear or disappear, or even change types. Consequently, the dependence on reference frame is an important challenge in the representation and analysis of vector fields. In particular, the fundamental question is how to determine the "correct" reference frame that exhibits the "correct" features.

In principle, this problem is equivalent to that of determining the "correct" reference frame for defining the laws of physics. Every experiment performed on Earth can be represented in many reference frames—the rotating reference frame of Earth, the revolving reference frame around the Sun, and so on. However, since an absolute ("correct") reference frame cannot be found, one may assume the reference frame used in a laboratory on Earth to be static, and derive physical laws in this frame. Such laws will be valid in infinitely many frames moving uniformly with respect to the laboratory.

Similar to these ideas in physics, the choice of the reference frames in the representation and analysis of vector fields is also subjective—there does not exist a single "correct" general-purpose reference frame. The choice of the reference frame is typically made out of convenience for performing the experiment or simulation; however, it generally cannot be guaranteed that the reference frame preferred for the experiment is also suitable for analysis. Figure 1.3 illustrates an example where the reference frame chosen for a flow simulation is not suitable to extract the desired rotational behavior—the von Kármán vortex street—from

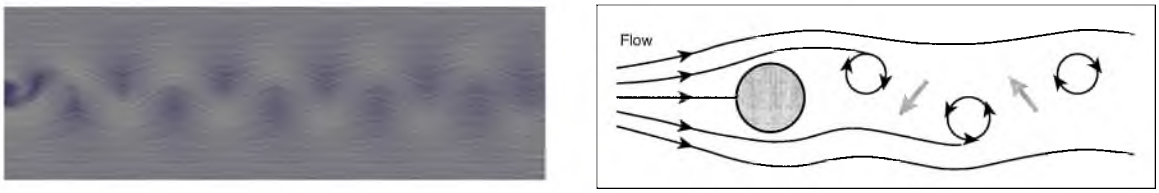


Figure 1.3. Streamline-based visualization of a simulated flow (left) fails to capture the expected rotational features (right), due to an unsuitable reference frame.

the flow. These features are especially important to identify in many engineering problems, such as the design of tall buildings and towers; failure to identify these structures correctly can lead to structure instabilities in design. Nevertheless, since there does not exist a "correct" reference frame, analysis tasks should define their own preferred reference frame to extract the features of interest, such as the von Kármán vortex street in the above example; this reference frame suitable for analysis may or may not be the same as the one suitable for simulation or experiment.

1.4.3 Challenges Due to the Temporal Dimension of Vector Fields

For unsteady flows, the features of interest are spatiotemporal in nature, meaning that they require spatial as well as temporal coordinates to describe their existence. Such features are typically extracted through analysis that is performed simultaneously in space and time. In particular, commonly used techniques are based on pathlines, which represent paths of material particles over time when advected in the flow. Typically, a large number of pathlines are traced for a finite amount of time; the pathlines are then used to extract important features from data. Examples of such analysis are the computation of pathline-based topology [203] and the detection of material boundaries using the finite-time Lyapunov exponent (FTLE) field [80].

However, such techniques are computationally expensive, since they must trace a large number of pathlines, of which typically only certain special ones are of importance. For example, a typical computation of the FTLE computes pathlines for every grid point in the domain. Furthermore, these pathlines are computed using numerical integration such as Runge-Kutta schemes, and require interpolation as well as integration of vectors in both space and time. As a result, they lack numerical robustness, and are affected by errors and uncertainties that are often nontrivial to track. Finally, they require simultaneous access to data across space and time, thus raising challenges of data management. In particular, even though pathline computation is parallelizable with respect to particles, the same spatiotemporal blocks of data may need to be loaded in and out of memory

multiple times, thus making them unattractive for large-scale data. Moreover, since pathline computation is temporally nonlocal, such techniques typically cannot be applied in situ using the data for a single time-step only. Especially considering the future of data analysis, where efficient and scalable techniques will be required to work at exascale, the approaches based on pathlines are not promising, despite the parallelization they offer.

In comparison, the analysis of time-varying scalar fields is much simpler: typically, features are extracted for instantaneous snapshots of data—that is, the field at each time-step—and these features are then tracked in time to understand their complete spatiotemporal behavior. However, it is well known that this model of analysis does not produce meaningful results for vector fields because the instantaneous features extracted from vector fields do not capture the underlying behavior in general. This limitation is closely related to the dependence upon reference frames, and it has been noted by many researchers that instantantaneous features are physically meaningful only under certain distinguished reference frames, but what these frames may be is not yet clear.

1.4.4 Challenges Due to Numerical Extraction of Dynamic Behavior

Vector fields are dynamic in nature; even small local perturbations in data (or in analysis) can have large global consequences. For example, even a small error in locating or advecting a particle in the field can cause it to flow to a significantly different region. Therefore, the computational challenges of numerical analysis are even more pronounced in the case of vector fields. In particular, the computation of streamlines—which, in the case of steady flows, represent the paths of massless particles traveling along the flow—is typically performed numerically using Runge-Kutta (RK) integration schemes [170]. The small errors incurred at each step of this numerical integration can accumulate and cause the computed streamline to deviate drastically from the "true" streamline.

Despite these problems, many of the standard techniques for vector field analysis rely on variants of RK methods. However, robust computation of flow becomes a formidable task, since the errors in numerical integration schemes can compound quickly. This lack of numerical robustness can create *inaccurate* and/or *inconsistent* analysis (as defined in Section 1.2.1), where any structures derived from streamlines—such as vector field topology [89], which is an abstract representation of the global flow behavior—are unreliable at best and possibly misleading at worst. Apart from the obvious problem of potentially including an incorrect structure in the analysis, numerical techniques can cause a more subtle yet equally important problem. Since the error in streamline tracing is nontrivial to capture, it is typically not shown. Hiding these inaccuracies and uncertainties inherent in

the numerical integration creates the perception of certainty. The user is presented with crisp streamlines and clean topological segmentations, implying a false sense of accuracy.

Furthermore, in the worst cases, these numerical errors in streamline computation may cause two distinct streamlines to intersect, as illustrated in Figure 1.4. Since streamlines are tangent curves to the vector field, an intersection implies two tangents at a single point—a physically invalid result that violates a fundamental property of vector fields that any two streamlines must be disjoint (that is, the uniqueness of the solution of an ordinary differential equation). The intersecting streamlines shown in Figure 1.4 are *separatrices* streamlines of saddle points. In 2D, separatrices represent separation and attachments lines [89, 106] and are crucial to detect for optimal designing of aircrafts, where flow separation and attachment along the aircraft wings can significantly increase the drag and raise its stall speed—conditions that are particularly dangerous during takeoff and landing. In this case, an inconsistent analysis can lead to faulty designs and structural failures, potentially causing catastrophic disasters.

Figure 1.5 illustrates another manifestation of inconsistency in vector field analysis that can be attributed to numerical techniques. When critical points are computed through numerical solutions of linear systems, there may exist false positives and false negatives. These results can violate the Poincaré index theorem (Theorem 2.3), which determines the

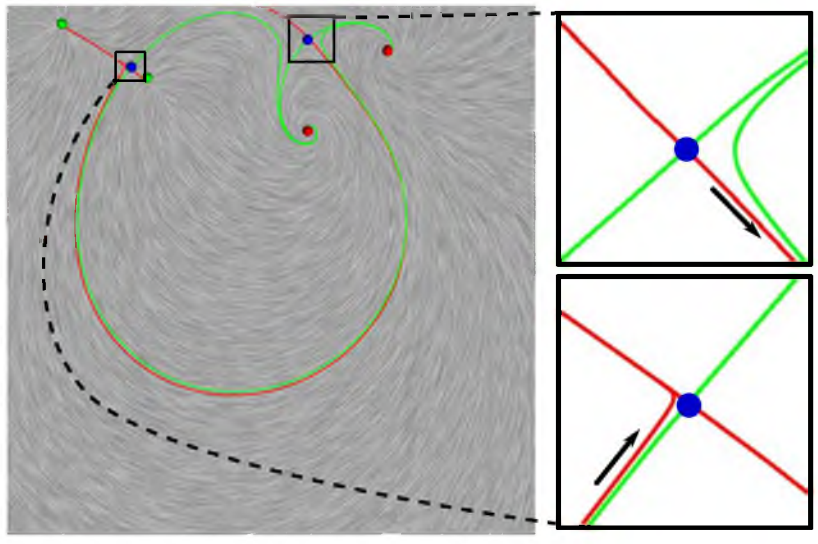


Figure 1.4. Inconsistent streamlines computed using numerical integration. With respect to the orientation of the red streamline marked with an arrow the green streamline switches from its left to its right, as the two streamlines travel counterclockwise (top zoom-in to bottom zoom-in). Thus, the accumulation of integration errors causes the two streamlines to intersect, creating inconsistent results.

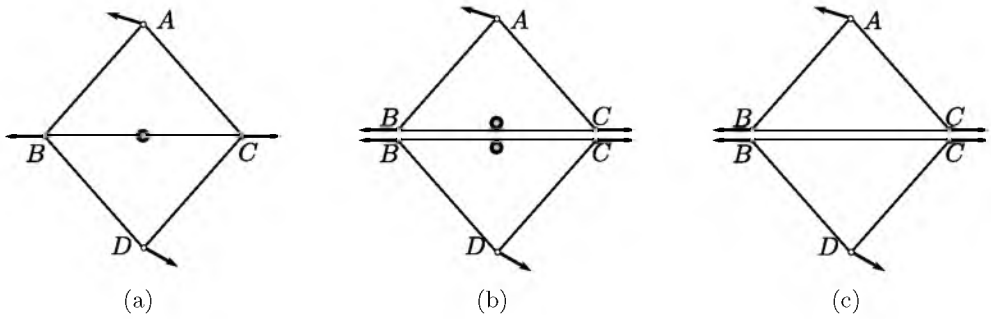


Figure 1.5. Inconsistent critical points detected using numerical techniques. (a) The actual configuration: a single critical points exists on the edge BC. Numerical techniques for identifying triangles containing critical points might test positive for (b) both ABC and BDC, or (c) neither of them—both of which are inconsistent results.

presence of critical points in a given region of a vector field—an inconsistency with severe consequences. For example, in the analysis of an MRI brain scan of an Alzheimer's patient, an important goal is to detect where the patient may be losing brain matter—a task that can be performed by detecting the critical points of the deformation vector field of the brain matter [126]. A false negative, in this scenario, can mislead the diagnosis, and ultimately prevent the treatment and understanding of the disease.

In general, computing the topological skeleton, which includes separatrices, critical points, and periodic orbits in the 2D case, can be numerically unstable due to errors inherent in the integration of separatrices and inconsistencies among neighboring cells of the domain. As a result, numerically computed structures may be adequate for visualization, but not suitable for further analysis.

1.5 Contributions of This Dissertation

The role of scientific data analysis is to enable the understanding of natural or artificial phenoemena by deriving important insights from the corresponding experiments and simulations. When a new theory or framework for analysis through feature extraction is developed, it must be able to address two particularly important questions:

Q1.) Are the extracted features correct?
Q2.) Is the extraction procedure precise?

The first question deals with the correctness of features in the sense of whether or not a feature must exist, or in what form it must exist, whereas the second question focuses on the precision of the analysis. This dissertation addresses these two questions in the context of vector fields. In particular, the contributions of this dissertation are divided into two

main parts: Part II develops new local reference frames to address the notion of correctness in feature extraction, and Part III develops new combinatorial representations that focus on the accuracy and consistency issues in analysis.

1.5.1 Thesis Statement

Features of vector fields cannot be defined objectively, but depend upon the assumed reference frame; therefore, analysis techniques must choose a suitable reference frame to focus on the features of interest. Appropriate reference frames can also reduce unsteady vector field analysis to the analysis of sequences of steady vector fields, which has important computational advantages. However, current techniques to perform such analysis are numerically nonrobust, and can produce results that are inconsistent with the underlying theory. Consistency is of prime importance in computational analysis, and must always be preserved, and new discrete theories and combinatorial representations can enable performing consistent analysis. This dissertation aims to develop new theories and frameworks to enable consistent feature extraction from vector fields.

1.5.2 Development of Local Reference Frames

As discussed in Section 1.4.2, the definition and choice of the "correct" reference frame for vector field analysis is subjective; therefore, the definition of the "correct" features is also not objective. As a result, this dissertation proposes that analysis applications define a preferred reference frame that is suitable to extract the features of interest.

The work presented in Part II of this dissertation defines a new reference frame useful for the analysis of a large class of simulated flows by focusing on localized features in the data. Thus, this part answers Question 1, asked above, by objectifying correctness in the context of the considered types of flows. An important advantage of the proposed reference frames is that they allow reducing unsteady flows to sequences of steady flows; that is, unsteady flows can be analyzed by treating the flow at individual time-steps as steady, and performing a feature extraction on each time-step. These extracted features can then be collated in time using a temporal analysis. For example, critical points and vortices can be computed independently for the flow at each time-step, and feature tracking can be performed to compute the paths of these features. As compared to pathline-based techniques, such analysis is less expensive, temporally local, allows better data management, and can be performed in situ. As a result, this work addresses many important mathematical and computational challenges discussed earlier.

In order to define this reference frame, **Chapter 3** develops a new flow decomposition

called the *natural Helmholtz-Hodge decomposition*, which eliminates an important limitation of the many existing formulations of the famous Helmholtz-Hodge decomposition (HHD). In particular, these conventional approaches impose boundary conditions to compute a unique HHD—a strategy associated with serious artifacts. Through the natural HHD, this dissertation eliminates these boundary artifacts by performing the decomposition in a natural and data-driven manner. In addition to being useful for defining a local reference frame, the natural HHD has many other advantages. It allows localized and multiscale flow analysis, and enables open-boundary analysis for scenarios where a boundary does not exist, or where the simulation does not impose any boundary conditions.

Chapter 4 uses the natural HHD to develop a new reference frame to highlight the localized properties of a given flow. This chapter demonstrates that this so-called *internal reference frame* is suitable for flows representing a large class of physical phenomena, as it enables the extraction of physically meaningful features that other techniques fail to detect. Finally, this chapter focuses on a very important application of the internal reference frame—the analysis of unsteady flows as sequences of steady flows. Case studies from various simulated flows demonstrate the applicability of the internal frame.

The research corresponding to Chapters 3 and 4 has been published as the following journal articles.

[1] Bhatia, H., Norgard G., Pascucci V., Bremer P.-T. Comments on the "Meshless Helmholtz-Hodge Decomposition". *IEEE Transactions on Visualization and Computer Graphics (TVCG) 19* (2013) 527–528. http://dx.doi.org/10.1109/TVCG.2012.62.

[2] Bhatia, H., Norgard G., Pascucci V., and Bremer P.-T. The Helmholtz-Hodge decomposition – A survey. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 19* (2013) 1386–1404. http://dx.doi.org/10.1109/TVCG.2012.316.

[3] Bhatia, H., Pascucci V., Kirby, R. M., and Bremer P.-T. Extracting features from time-dependent vector fields using internal reference frames. *Computer Graphics Forum 33* (2014), 21–30. http://dx.doi.org/10.1111/cgf.12358.

[4] Bhatia, H., Pascucci V., and Bremer P.-T. The natural Helmholtz-Hodge decomposition for open-boundary flow analysis. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 20* (2014) 1566–1578. http://dx.doi.org/10.1109/TVCG.2014.2312012.

As a motivation to develop new techniques for analyzing unsteady flows, **Chapter 5** describes some work in progress. In particular, a detailed case study is provided that compares the pathlines computed using state-of-the-art techniques with particles propagated in

situ in a combustion simulation. This quantitative and qualitative evaluation demonstrates that, in practice, pathlines computed for large-scale complex simulated flows do not capture the true particle behavior, which calls into question the widely applied analysis model using pathlines.

1.5.3 Development of Combinatorial Representations

As discussed in Section 1.2.1, computational models for experiments, simulations, and data analysis are typically based upon mathematical theories that assume smooth fields defined over smooth domains. However, practical adaptations of these theories are numerical in nature, and are fundamentally limited in how the data is represented and computations are performed. This analysis model typically incurs large amounts of errors, which, more importantly, are nontrivial to track.

To address the issues caused by the lack of numerical robustness in vector field analysis, Part III of this dissertation develops a new discrete theory of representation and analysis of vector fields. As compared to smooth theory, the primary advantage of the discrete world is that its data and concepts can be adapted for computers exactly using combinatorial representations. The primary goal of this work is to guarantee consistency in feature extraction for vector fields, and at the same time, track the uncertainties in representation and analysis. New representations and algorithms developed in Part III achieve this goal, and answer Question 2 asked above to lend credibility to analysis.

Chapter 6 introduces *Edge Maps*, which define a new theory of representation for 2D piecewise linear (PL) vector fields. They are an explicit representation of flow as they encode the most fundamental property—streamlines—needed by analysis, and eliminate the need for numerical integration of streamlines. Using Edge Maps, which can also encode the discretization errors incurred during their construction, errors in streamline tracing can be tracked to provide candid uncertainty visualizations. This chapter discusses the construction and properties of Edge Maps in detail, and provides an exhaustive list of equivalence classes of Edge Maps for PL vector fields.

Chapter 7 introduces a quantized version of Edge Maps, which eliminates the need for floating-point arithmetic by using integer arithmetic to compute streamlines. Thus, these *quantized Edge Maps* define a new discrete theory based upon the PL theory developed by Edge Maps. Using the quantized Edge Maps, all possible streamlines in the flow can be encoded into a combinatorial datastructure through an integer-based representation. This combinatorial representation of vector fields allows guaranteeing consistency during

streamline tracing. This chapter also introduces a new combinatorial algorithm to perform consistent critical point detection in both 2D and 3D.

The research corresponding to Chapters 6 and 7 has been published as the following journal articles, conference proceedings, and book chapters.

[5] Bhatia, H., Jadhav S., Bremer P.-T., Chen G., Levine J. A., Nonato L. G., and Pascucci V. Edge maps: Representing flow with bounded error. In *Proceedings of the 4th IEEE Pacific Visualization Symposium* (2011) pp. 75–82, Hong Kong, China. http://dx.doi.org/10.1109/PACIFICVIS.2011.5742375.

[6] Jadhav S., Bhatia, H., Bremer P.-T., Levine J. A., Nonato L. G., and Pascucci V. Consistent approximation of local flow behavior for 2D vector fields. In *Topological Methods in Data Analysis and Visualization II – Theory, Algorithms, and Applications*, R. Peikert, H. Hauser, H. Carr, and R. Fuchs, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2012, pp. 141–159. http://dx.doi.org/10.1007/978-3-642-23175-9_10.

[7] Bhatia, H., Jadhav S., Bremer P.-T., Chen G., Levine J. A., Nonato L. G., and Pascucci V. Flow visualization with quantified spatial and temporal errors using edge maps. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 18* (2012) 1386–1396. http://dx.doi.org/10.1109/TVCG.2011.265.

[8] Levine J. A., Jadhav S., Bhatia, H., Pascucci V., and Bremer P.-T. A quantized boundary representation of 2D flow. *Computer Graphics Forum 31* (2012), 945–954. http://dx.doi.org/10.1111/j.1467-8659.2012.03087.x.

[9] Bhatia, H., Gyulassy A., Wang H., Bremer P.-T., and Pascucci V. Robust detection of singularities in vector fields. In *Topological Methods in Data Analysis and Visualization III – Theory, Algorithms, and Applications*, P.-T. Bremer, I. Hotz, V. Pascucci, R. Peikert, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2014, pp. 3–18. http://dx.doi.org/10.1007/978-3-319-04099-8_1.

PART I

BACKGROUND

CHAPTER 2

FUNDAMENTALS AND RELATED WORK

2.1 Potential Theory

In order to understand the concepts presented in Chapter 3, it is important to have a clear understanding of the potential theory. This section presents some of its fundamental elements; a more detailed discussion can be found in standard textbooks [7, 135].

2.1.1 Poisson and Laplace Equations

The *Poisson equation* is one of the most fundamental elliptic partial differential equations, and is expressed as

$$\nabla^2 \varphi = f \qquad \text{in } \Omega, \tag{2.1}$$

where f is called the *source function* and φ the *potential function*. The general solution to Equation (2.1) is given by $\varphi = F + H$, where $\nabla^2 F = f$, and H is a solution of the *Laplace equation*,

$$\nabla^2 H = 0 \qquad \text{in } \Omega, \tag{2.2}$$

and is called a *harmonic function*. It is important to note that one can add any harmonic function H to φ while still satisfying Equation (2.1). To obtain a *particular solution* to the Poisson equation (2.1), one must choose a specific H, typically done by imposing boundary conditions.

In physical terms (refer to Figure 2.1), the Poisson equation (2.1) solves for the potential φ generated by the (energy) sources *inside* the domain (defined by f), as well as those *outside* the domain. However, since for the Laplace equation (2.2), the (energy) sources are zero inside the domain, its solution (a harmonic function) represents the *external influence* [7, 135], that is, the potential generated by the (energy) sources lying outside Ω. The external influence can be inferred if its boundary values are known. Therefore, the harmonic function representing the external influence is completely and uniquely defined by boundary conditions. Since the *internal influence* is already uniquely defined by f, when

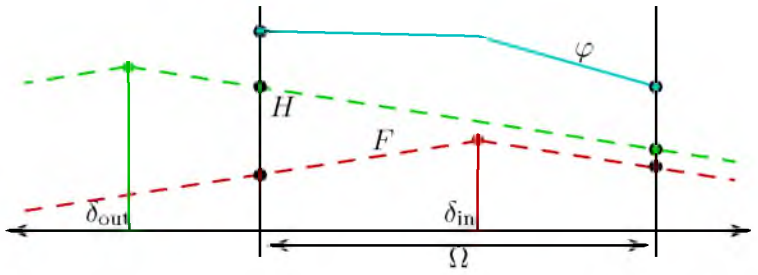


Figure 2.1. The Poisson equation computes the potential generated by specified energy sources, with respect to the given boundary conditions. The figure illustrates the Poisson equation on a 1D domain $\Omega \subset \mathbb{R}$. Inside Ω, the potential due to an internal source δ_{in} (red) is uniquely defined. However, the potential due to any external source, for example, δ_{out} (green) is harmonic (constant slope) inside Ω, and can be determined uniquely using boundary values. Therefore, the solution to the Poisson equation (cyan)—the potential due to interior and exterior sources—is uniquely defined using the boundary conditions.

appropriate boundary conditions are given, φ (the solution to Equation (2.1)) is uniquely defined.

2.1.1.1 Boundary conditions. Two common types of boundary conditions to solve the Poisson equation (2.1) are

- *Dirichlet*, which specifies the value of φ at the boundary, that is, $\varphi = g$ on $\partial\Omega$. In this case, there exists a unique solution.
- *Neumann*, which specifies the value of the derivative of ϕ with respect to the boundary, that is, $\frac{\partial\varphi}{\partial\vec{n}} = g$ on $\partial\Omega$. In this case, there exists a solution unique up to an additive constant if and only if the following compatibility condition is satisfied [36].

$$\int_{\Omega} f \ \mathrm{d}V = \int_{\partial\Omega} g \ \mathrm{d}A. \tag{2.3}$$

The additive constant can be removed by requiring that $\varphi(\mathbf{x}_0) = 0$, $\mathbf{x}_0 \in \Omega$.

2.1.1.2 Green's function. One way to solve the Poisson and Laplace equations is to use an integral kernel called the *Green's function*, $\mathrm{G}(\mathbf{x}, \mathbf{x}_0)$. The Green's function is defined as the potential created at $\mathbf{x}_0$ due to an impulse (energy) source represented by the *Dirac delta function* located at $\mathbf{x}$ (as shown in Figure 2.1), that is, it is the solution to the equation

$$\nabla^2 \mathrm{G}(\mathbf{x}, \mathbf{x}_0) = \delta(\mathbf{x} - \mathbf{x}_0).$$

In the case of a single source located at $\mathbf{x}$ in an infinite domain, it is also called the *free-space Green's function*, $\mathrm{G}_\infty = \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)$, and is given as

$$\begin{aligned} \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) &= \frac{1}{2}|\mathbf{x}-\mathbf{x}_0| && \mathbf{x},\mathbf{x}_0 \in \mathbb{R}, \\ \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) &= \frac{1}{2\pi}\log(|\mathbf{x}-\mathbf{x}_0|) && \mathbf{x},\mathbf{x}_0 \in \mathbb{R}^2, \\ \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) &= \frac{-1}{4\pi|\mathbf{x}-\mathbf{x}_0|} && \mathbf{x},\mathbf{x}_0 \in \mathbb{R}^3. \end{aligned} \tag{2.4}$$

According to the *sifting property* of the Dirac delta function, any continuous source function can be represented as a collection of infinitely many impulse sources, as

$$f(\mathbf{x}_0) = \int_{\mathbb{R}^n} f(\mathbf{x})\ \delta(\mathbf{x}-\mathbf{x}_0)\,\mathrm{d}\mathbf{x}.$$

Then, the Poisson equation $\nabla^2\varphi = f$ on $\mathbb{R}^n$, with $f(\mathbf{x}) \to 0$ for $\mathbf{x} \to \infty$ can be solved as

$$\varphi(\mathbf{x}_0) = \int_{\mathbb{R}^n} \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ f(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{2.5}$$

The above *integral solution* [135] computes the potential φ at a point $\mathbf{x}_0 \in \mathbb{R}^n$ due to the sources defined by f at all $\mathbf{x} \in \mathbb{R}^n$.

To compute φ for a bounded domain Ω using the integral solution, one typically uses the *finite-space Green's function*, G_Ω, which is influenced by both the domain Ω and its boundary $\partial\Omega$. In this case, the Poisson equation (2.1) can be solved [135] as

$$\varphi(\mathbf{x}_0) = \int_\Omega \mathrm{G}_\Omega\ f(\mathbf{x})\,\mathrm{d}\mathbf{x} + \oint_{\partial\Omega} \varphi(\mathbf{x})\frac{\partial \mathrm{G}_\Omega}{\partial\vec{n}}\,\mathrm{d}\mathbf{x} - \oint_{\partial\Omega} \mathrm{G}_\Omega\frac{\partial\varphi(\mathbf{x})}{\partial\vec{n}}\,\mathrm{d}\mathbf{x}.$$

In the case of Dirichlet boundary conditions, one typically finds a G_Ω that vanishes at the boundary, such that the third integral becomes zero. On the other hand, for Neumann boundary conditions, one imposes $\frac{\partial \mathrm{G}_\Omega(\mathbf{x},\mathbf{x}_0)}{\partial\vec{n}} = 1/\left(\oint_{\partial\Omega}\mathrm{d}\mathbf{x}\right)$, in which case the second integral becomes a constant. In either case, the general form of G_Ω is given as

$$\mathrm{G}_\Omega(\mathbf{x},\mathbf{x}_0) = \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) + \gamma_\Omega(\mathbf{x},\mathbf{x}_0),$$

such that, $\mathrm{G}_\Omega(\mathbf{x},\mathbf{x}_0) = 0$ for $\mathbf{x} \in \partial\Omega$, and γ_Ω is harmonic on Ω. Thus, computing the Green's function for an arbitrary domain is nontrivial, and typically, it is not available in closed-form.

2.2 Degree Theory

The consistent critical point detection technique presented in Section 7.2 relies on certain elements of degree theory, which will be used to prove consistency. The definitions presented in this section have been made by Lloyd [123].

In the context of this discussion, let Ω be a bounded, open subset of $\mathbb{R}^n$. The closure and the boundary of Ω are denoted by $\bar{\Omega}$ and $\partial\Omega$, respectively. A point $\mathbf{x} \in \mathbb{R}^n$ is denoted

as $\mathbf{x} = (x_0, ..., x_{n-1})$, and has an L_∞ norm $|\mathbf{x}| = \max_{(0 \le i \le n-1)}\{|x_i|\}$. $C(\bar{\Omega})$ denotes the class of continuous functions $\varphi \in C(\bar{\Omega})$, such that $\varphi(\mathbf{x}) : \bar{\Omega} \to \mathbb{R}^n$ with the norm $||\varphi|| = \sup_{\mathbf{x}\in\Omega} |\varphi(\mathbf{x})|$. $C^1(\bar{\Omega})$ is a subset of $C(\bar{\Omega})$ such that $\varphi \in C^1(\bar{\Omega})$ has continuous first-order partial derivatives. Let $\mathbf{p} = \varphi(\mathbf{x}) = (\varphi_0(\mathbf{x}), \cdots, \varphi_{n-1}(\mathbf{x}))$; then, the Jacobian matrix $\mathfrak{J}$ of φ is given as $\mathfrak{J}\varphi(\mathbf{x}) = [\nabla\varphi_0, \cdots, \nabla\varphi_{n-1}]^T$. For $\varphi \in C^1(\bar{\Omega})$, $\mathbf{p} = \varphi(\mathbf{x})$ is called a *degenerate value* of φ if there exists $\mathbf{x} \in \bar{\Omega}$ such that $\det(\mathfrak{J}\varphi(\mathbf{x})) = \mathbf{0}$; otherwise $\mathbf{p}$ is a *regular value* of φ.

For a bounded, open subset $\Omega \subset \mathbb{R}^n$, and a continuous function $\varphi \in C(\bar{\Omega})$, the Brouwer degree of φ in Ω with respect to the value $\mathbf{p}$, where $\mathbf{p} \notin \varphi(\partial\Omega)$, is defined as follows:

Definition 2.1 (Brouwer Degree for $C^1(\bar{\Omega})$ and regular value $\mathbf{p}$)**.** If $\varphi \in C^1(\partial\Omega)$ and $\mathbf{p}$ is a regular value of φ, then

$$\deg(\varphi, \Omega, \mathbf{p}) = \sum_{\mathbf{x}\in\varphi^{-1}(\mathbf{p})} \operatorname{sign}\left(\det(\mathfrak{J}\varphi(\mathbf{x}))\right).$$

An intuition behind the concept of Brouwer degree is illustrated in Figure 2.2(a). It is essentially the number of the net crossings of $\mathbf{p}$ by the image of $\partial\Omega$ under φ. The above definition is limited to regular values $\mathbf{p}$ only. Not all values in a sampled function reconstructed through interpolation are regular, so our definition must encompass degenerate values as well.

Definition 2.2 (Brouwer Degree for $C^1(\partial\Omega)$ and degenerate value $\mathbf{p}$)**.** If $\varphi \in C^1(\partial\Omega)$ and $\mathbf{p}$ is a degenerate value of φ, then

$$\deg(\varphi, \Omega, \mathbf{p}) = \deg(\varphi, \Omega, \mathbf{p}_1),$$

where $\mathbf{p}_1$ is any regular value such that $|\mathbf{p} - \mathbf{p}_1| < \operatorname{dist}(\mathbf{p}, \varphi(\partial\Omega))$.

The existence of a nondegenerate value in every neighborhood of $\mathbf{p}$ is guaranteed by Sard's Theorem [123]. Similarly, we would like to extend this definition to functions that are continuous, but not necessarily in $C^1(\partial\Omega)$.

Definition 2.3 (Brouwer Degree for $C(\partial\Omega)$)**.** If $\varphi \in C(\partial\Omega)$, then

$$\deg(\varphi, \Omega, \mathbf{p}) = \deg(\varphi_1, \Omega, \mathbf{p}),$$

where φ_1 is any function in $C^1(\partial\Omega)$ such that $|\varphi(\mathbf{x}) - \varphi_1(\mathbf{x})| < \operatorname{dist}(\mathbf{p}, \varphi(\partial\Omega)$, for any $\mathbf{x} \in \partial\Omega$.

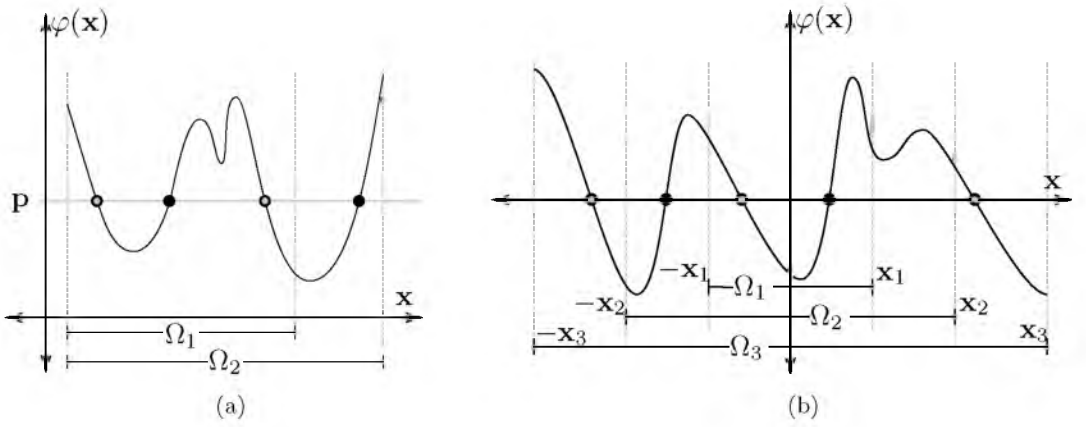


Figure 2.2. The Brouwer degree counts the net number of times a function value $\mathbf{p}$ is crossed by the image of the closure $\bar{\Omega}$ of a domain under function φ. The "positive" ($\det(\mathfrak{J}\varphi(\mathbf{x})) > 0$) and "negative" ($\det(\mathfrak{J}\varphi(\mathbf{x})) < 0$) crossings are shown as solid and hollow dots, respectively. (a) For the open and bounded sets Ω_1 and Ω_2, the Brouwer degree of $\varphi(\mathbf{x})$ with respect to the value $\mathbf{p}$ is -1 and 0, respectively. (b) When the sets Ω_i are symmetric, Theorem 2.2 guarantees that if the values on the boundary have different signs, then the Brouwer degree with respect to $\mathbf{0}$ is odd, as is the case for Ω_2 and Ω_3.

Basically, one can find a function φ_1 that is "close" to φ and has continuous derivatives, and define the Brouwer degree with respect to this function.

Using the Brouwer degree, the net number of crossings of $\mathbf{p}$ by the image of $\bar{\Omega}$ can be counted. If this number is nonzero, then there exists at least one $\mathbf{x}$ such that $\varphi(\mathbf{x}) = \mathbf{p}$, which leads to the following theorems:

Theorem 2.1 (Kronecker's Existence Theorem)**.** If $\deg(\varphi, \Omega, \mathbf{p}) \neq 0$, then the equation $\varphi(\mathbf{x}) = \mathbf{p}$ has at least one solution in Ω.

Theorem 2.2. Let Ω be a bounded, open, symmetric subset of $\mathbb{R}^n$ containing the origin. If $\varphi : \bar{\Omega} \to \mathbb{R}^n$ is continuous, $\mathbf{0} \notin \varphi(\partial\Omega)$, and for all $\mathbf{x} \in \partial\Omega$

$$\frac{\varphi(\mathbf{x})}{|\varphi(\mathbf{x})|} \neq \frac{\varphi(-\mathbf{x})}{|\varphi(-\mathbf{x})|},$$

then $\deg(\varphi, \Omega, \mathbf{0})$ is an odd number [123].

Intuitively, Theorem 2.2 ensures that φ crosses $\varphi(\mathbf{x}) = 0$ at least once if no antipodal vectors of φ are parallel, as can be seen in Figure 2.2(b).

2.3 Vector Fields

For a given nD domain $\Omega \subseteq \mathbb{R}^n$, a map that assigns each point in space a vector is called a *vector field*, that is, $\vec{V} : \Omega \to \mathbb{R}^n$. For a given point $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Omega$, the

vector field $\vec{V}(\mathbf{x})$ consists of n scalar functions determining the value of each component $\vec{V}(\mathbf{x}) = \left(\vec{V}_1, \vec{V}_2, \ldots, \vec{V}_n\right)$. Figure 2.3 illustrates a 2D vector field.

For 2D or 3D vector fields, the components of the vector field are typically referred to as $\vec{V} = (u, v)$ or $\vec{V} = (u, v, w)$, respectively. Given a differentiable manifold $\mathbb{M}$, a vector field on $\mathbb{M}$ assigns a tangent vector to each point in $\mathbf{x}$. That is, $\vec{V} : \mathbb{M} \to \mathbb{TM}$, where $\mathbb{TM}$ is the tangent space of $\mathbb{M}$. The vector field as defined above depends only upon space. Such a vector field that does not change with time is called a *steady* vector field. In comparison, an *unsteady* vector field $\vec{V} : \mathbb{R} \times \Omega \to \mathbb{R}^n$ or $\vec{V} : \mathbb{R} \times \mathbb{M} \to \mathbb{TM}$ varies with time, and is denoted as $\vec{V}(t, \mathbf{x})$.

2.3.1 Differential Operations on Vector Fields

One of the most fundamental differential operators for vector fields is the *nabla* operator and is expressed as

$$\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right)^T,$$

where $(x_1, \ldots, x_n)$ are the coordinates representing a point in $\mathbb{R}^n$. Using the nabla operator, various operations can be performed on vector fields.

2.3.1.1 Divergence. The divergence ($\nabla \cdot$) measures the amount of expansion or contraction of the flow at a given point, or the creation or destruction of material. The presence of divergence reflects compressibility in the flow, and is expressed as the scalar (dot) product of the nabla operator with the vector field.

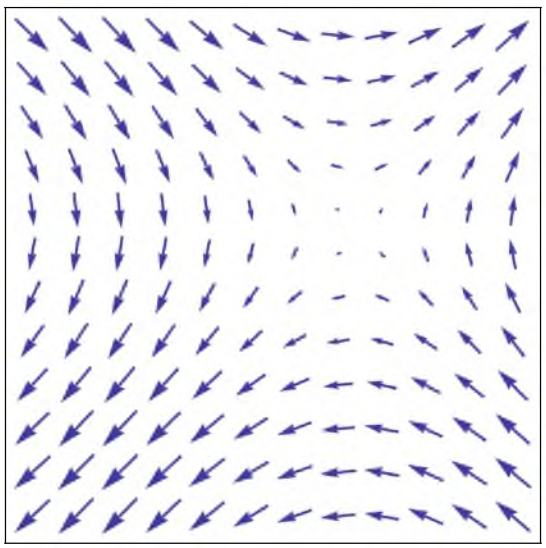


Figure 2.3. A 2D steady vector field. Each point in the domain is assigned a vector, that is, a magnitude and a direction, visualized as arrows.

$$\text{div } \vec{V}(\mathbf{x}) = \nabla \cdot \vec{V}(\mathbf{x}) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}.$$

For a given domain Ω with boundary $\partial\Omega$, the total amount of flow created and destroyed inside the domain must match the sum of the influx and outflux in the domain. This result is commonly known as the *Gauss theorem*, or the *divergence theorem* [71].

$$\int_{\Omega} \nabla \cdot \vec{V} \, \mathrm{d}\mathbf{x} = \oint_{\partial\Omega} \vec{n} \cdot \vec{V} \, \mathrm{d}\mathbf{x}.$$

where $\vec{n}$ is the outward normal to the boundary.

2.3.1.2 Curl. The curl ($\nabla\times$) measures the rotation of the flow about an instantaneous axis at a given point. The curl vector indicates the axis and magnitude of the rotation, and is expressed as the vector (cross) product of the nabla operator with the vector field.

$$\text{curl } \vec{V}(\mathbf{x}) = \nabla \times \vec{V}(\mathbf{x}) = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial x}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial y}{\partial x} - \frac{\partial x}{\partial y} \right).$$

Similar to the divergence theorem, the *curl theorem* relates the rotation of the vector field in Ω to the vectors at the $\partial\Omega$. In particular, the curl theorem says that the total rotation of the flow inside the domain must match the integrated tangential component of the flow at the boundary.

$$\int_{\Omega} \nabla \times \vec{V} \, \mathrm{d}\mathbf{x} = \oint_{\partial\Omega} \vec{n} \times \vec{V} \, \mathrm{d}\mathbf{x}.$$

The terms *rotation* and *vorticity* are also used to express the curl. In the 2D case, the axis of rotation is always normal to the plane, and, therefore, one can assume the direction, and represent the curl simply by a scalar given below.

$$\text{curl } \vec{V}(\mathbf{x}) = \nabla \times \vec{V}(\mathbf{x}) = \frac{\partial y}{\partial x} - \frac{\partial x}{\partial y}.$$

2.3.1.3 Gradient. The gradient of a vector field $\vec{V}$ is the matrix of its first-order partial derivatives, and is called the *deformation tensor* or the *Jacobian*, $\mathfrak{J}\vec{V}$, of the field.

$$\mathfrak{J}\vec{V}(\mathbf{x}) = \nabla \vec{V} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix}.$$

2.3.2 Special Types of Vector Fields

Depending upon the differential operators discussed above, some special vector fields are defined below.

- An *incompressible* vector field is a vector field $\vec{V}$ such that its divergence is zero at all points in the domain, that is, $\nabla \cdot \vec{V}(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega$. Such vector fields are also called *divergence-free* or *solenoidal* vector fields, and are used to represent flow of material that does not compress, for example, water.

- An *irrotational* vector field is a vector field $\vec{V}$ such that its curl is zero at all points in the domain, that is, $\nabla \times \vec{V}(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega$. Such vector fields are also called *rotation-free* vector fields, and are used to represent gradients of scalar fields.

- A *harmonic* vector field is a vector field that is both incompressible and irrotational, that is, $\nabla \cdot \vec{V}(\mathbf{x}) = 0$ and $\nabla \times \vec{V}(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega$. Harmonic vector fields will play an important role in the later parts of this work, and will be discussed in more detail in Chapter 3.

2.3.3 Characteristic Curves of Vector Fields

An important way of describing vector fields is through certain integral lines that highlight different properties of the field. In the following, we will discuss only two most relevant types of curves—streamlines and pathlines. Description of other curves—streaklines and timelines—can be found in any standard text on vector fields.

2.3.3.1 Streamlines. These are the curves that are tangent to the vector field at every point, and, thus, show the direction that a fluid element will take at any point in time. Let $\mathbf{x}_S(s)$ denote the parametric representation of a single streamline. Then it is defined as

$$\frac{\mathrm{d}\mathbf{x}_S(s)}{\mathrm{d}s} \times \vec{V}(\mathbf{x}_S(s)) = \vec{0}. \tag{2.6}$$

Since the streamlines are always parallel to the vector field, the slope of any streamline matches that of the vector field. Given the components of the vector field, $\vec{V} = (u, v, w)$, a streamline $\mathbf{x}_S(s) = (x, y, z)$ satisfies

$$\frac{\mathrm{d}x}{u} = \frac{\mathrm{d}y}{v} = \frac{\mathrm{d}z}{w}. \tag{2.7}$$

Streamlines are defined for instantaneous velocities, and, therefore, time (t) plays no role in their definition, and is generally assumed. Furthermore, by definition, different streamlines at the same instant in a flow do not intersect, because a fluid particle cannot have two different velocities at the same point.

2.3.3.2 Pathlines. These are the trajectories that massless particles follow, when advected under the flow over a certain period of time. Intuitively, a pathline can be seen as

"tagging" a particle in the flow, and taking a long exposure photograph of its motion. Let $\mathbf{x}_P(t)$ denote the parametric representation of a pathline. Then, it is defined as

$$\begin{aligned} \frac{d\mathbf{x}_P(t)}{dt} &= \vec{V}(t, \mathbf{x}_P(t)), \\ \mathbf{x}_P(t_0) &= \mathbf{x}_{P0}. \end{aligned} \tag{2.8}$$

Note that t_0 will be a fixed value, and given a particular t_0, the corresponding pathline $\mathbf{x}_P(t)$ gives a parametric equation in t, that is, by varying t, we can compute the position of the particle at any instant of time. Also note that the direction the path takes will be determined by the streamlines of the fluid at each moment in time.

2.3.3.3 Streamlines versus pathlines. For unsteady vector fields, streamlines and pathlines are distinct, since streamlines represent instantaneously tangent direction, and pathlines represent time-integrated paths. However, in the case of steady vector fields, streamlines and pathlines coincide, that is, $\mathbf{x}_S(s) = \mathbf{x}_P(t)$ when the same parameter is used, that is, $s = t$. For such cases, streamlines can also be defined as the paths of massless particles traced under the steady field, and computed as numerical integration similar to pathlines. Consequently, when the context is clear, the subscript will be eliminated.

2.3.4 Topological Characteristics of Vector Fields

This section introduces some important topological characteristics of vector fields, which can be found in more detail in any standard text book on topology [92].

2.3.4.1 Winding number. Consider a continuous vector field $\vec{V}$ and a closed curve γ, such that there are no critical points of $\vec{V}$ on γ. Let $\mathbf{p}$ be a point on γ that is moved counterclockwise along the curve. As the point is moved, the vector $\vec{V}(\mathbf{p})$ changes values, and makes a whole number of rotations before coming back to the initial position. This number of rotations is called the *winding number* of $\vec{V}$ with respect to γ. The winding number is positive if $\vec{V}$ makes a net counterclockwise rotation and negative if the net rotation is clockwise. The winding number contains information about the behavior of $\vec{V}$ in the region contained inside γ.

2.3.4.2 Isolated critical points. *Critical points* are the zeros of a vector field, that is, $\vec{V}(\mathbf{c}) = 0$ implies that $\mathbf{c}$ is a critical point. For a given critical point $\mathbf{p}$, if there exists a neighborhood that does not contain another critical point, then $\mathbf{c}$ is called an *isolated critical point*. The winding number can be used to study isolated critical points by considering a curve that bounds a small neighborhood around them. In particular, the *index* of an isolated critical point $\mathbf{c}$ of $\vec{V}$ is the winding number of a small counterclockwise oriented curve $\gamma_\mathbf{c}$ centered around $\mathbf{c}$ such that $\gamma_\mathbf{c}$ contains no other critical points.

2.3.4.3 Types of critical points. A critical point is called a *center* if there exist closed streamlines that swirl around it, and no streamline passes through it. A *node* is a critical point where nearby streamlines end, whereas a *focus* is the one around which the streamlines swirl endlessly, but never reach the critical point. A *saddle* is a critical point where exactly four streamlines meet—two outgoing and two incoming. Note that, technically, no streamline ever reaches a critical point since the vector at a critical point is zero, and so a tangent is not defined. Nevertheless, a streamline is informally considered to end/begin at a critical point when it gets infinitesimally close to the critical point. For a hyperbolic critical point (nodes and foci), or a center, the index is +1, whereas for a saddle, the index is -1.

2.3.4.4 The Poincaré index. The index of a critical point is also commonly called its *Poincaré index*. Using the concept of indices, it is possible to determine the behavior of a vector field enclosed by a closed curve γ by observing its behavior on γ.

Theorem 2.3 (The Poincaré Index Theorem)**.** Consider a continuous vector field $\vec{V}$ and a closed curve γ, such that there are no critical points of $\vec{V}$ on γ. The winding number of $\vec{V}$ with respect to γ equals the sum of the indices of the critical points lying inside the domain bounded by γ.

The work published by Henri Poincaré in 1881 was generalized by Heinz Hopf in 1926. A direct consequence of the work of Poincaré and Hopf is the fact that there cannot exist a vector field on a sphere without any critical points. This is an important topological invariant that connects the behavior of a vector field in terms of its critical points to the topology of the underlying domain.

2.3.5 Eulerian and Lagrangian Representations

The behavior of a flow can be represented in two ways: Eulerian and Lagrangian. This section briefly describes the two representations, and for more detail, the reader can refer to the book by Panton [157] and the lecture notes by McDonough [138].

In the *Eulerian* representation, the flow is defined as a field; the properties of the field—vector values, divergence, rotation, etc.—are defined at fixed coordinates in space and time. For example, to understand the wind patterns around Earth's surface, one can build a grid of weather stations at fixed positions on Earth's surface, and measure the wind vectors at the points. The resulting data, $\vec{V}(t, \mathbf{x})$, is an Eulerian representation of the wind vector field on Earth.

On the other hand, the *Lagrangian* representation assigns the dynamic properties of the flow to moving fluid particles; the equations are given for different particles, instead of defining them for space and time. For example, the goal of understanding wind patterns can also be attained using a registering balloon that is released at a certain position, and is advected by the flow. The Lagrangian specification $\mathbf{X}(t, \mathfrak{a})$ describes the position of a given particle $\mathfrak{a}$ at all times—its pathline. The data captured by the balloon is given in the Lagrangian representation.

Both representations are equivalent, and one can be transformed into the other using the following expression:

$$\vec{V}(t, \mathbf{X}(\mathfrak{a}, t)) = \frac{\partial \mathbf{X}}{\partial t}(t, \mathfrak{a}) . \tag{2.9}$$

2.3.5.1 Material derivative. The Lagrangian and Eulerian specifications of the kinematics and dynamics of the flow are related by the *material derivative*.

$$\frac{\mathrm{D}f}{\mathrm{D}t} = \frac{\partial f}{\partial t} + \left(\vec{V} \cdot \nabla\right) f. \tag{2.10}$$

The material derivative describes how a certain property of the flow, $f(t, \mathbf{x})$, changes along the path of a particle over time. The material derivative can be applied to scalar, vector, or tensor quantities.

2.3.6 Reference Frames

The Eulerian representation of flow has to be evaluated in a certain *reference frame*, which defines the assumed coordinate system to measure the field. Due to this dependence, the Eulerian or point-based properties of the flow are not invariant to different reference frames. The choice of the reference frame is often guided by the suitability of experimental setup, or accessibility to certain frames. Typically, there is no distinguished reference frame, and different reference frames can be transformed into each other.

2.3.6.1 Galilean invariance. Galilean transformations are an important subset of the possible transformations between different reference frames. Let the flow properties of $\vec{V}(t, \mathbf{x})$ be defined with respect to a given reference frame $\mathcal{F}$. If the observer starts moving at a uniform velocity $\vec{u}(t, \mathbf{x}) = \vec{u}$ with respect to $\mathcal{F}$, then a different reference frame $\mathcal{F}'$ is obtained. The coordinate system of this new reference frame, $\mathbf{x}'$, is related to that of $\mathcal{F}$ as follows:

$$\mathbf{x}' = \mathbf{x} + \vec{u}\, t,$$

where the time t is the same in both reference frames (as in Newtonian relativity). The flow velocity observed in $\mathcal{F}'$ is given as

$$\begin{aligned}\vec{V}(t, \mathbf{x}') &= \vec{V}(t, \mathbf{x}) - \vec{u}, \\ &= \vec{V}(t, \mathbf{x}' + \vec{u}\ t) - \vec{u}.\end{aligned}$$

This uniform translation between $\mathcal{F}$ and $\mathcal{F}'$ is called a *Galilean transformation*. Such transformations are particularly important since a uniform motion has zero acceleration, and does not add any external forces to the system (force $\propto$ acceleration).

A property of the flow—scalar, vector, or tensor—is *Galilean invariant*, if it does not change under a Galilean transformation. For example, the velocity is changed under a Galilean transformation as can be seen above, and is, therefore, not Galilean invariant. On the other side, the Jacobian of a flow ignores uniform motion, and is, therefore, Galilean invariant. The notion of reference frames and invariances will be revisited in Chapter 4.

2.4 Discrete Representations of Smooth Fields

Practical applications do not have access to smooth functions. Instead, they have to work with discrete nonsmooth functions that approximate smooth functions. Specialized representations must be developed to improve the quality of these approximations. This section discusses important discrete representations for domains and vector fields along with an important way of removing ambiguities that may arise in such representations.

2.4.1 Simplicial Complexes

Simplicial complexes are an important datastructure used to define discrete functions by representing topological spaces as triangular and tetrahedral meshes. This section presents some important definitions required to construct simplicial complexes [47].

Definition 2.4 (Affine Combination). Given a set of $k+1$ points $\{\mathbf{p}_0, \mathbf{p}_1, \ldots \mathbf{p}_k\}$, with $\mathbf{p}_i \in \mathbb{R}^n$, a point $\mathbf{x} = \sum_{i=0}^{k} \lambda_i \mathbf{p}_i$ is an *affine combination* of $\{\mathbf{p}_i\}$ if $\sum_{i=0}^{k} \lambda_i = 1$.

The points $\{\mathbf{v}_i\}$ are *affinely independent* if $\sum \lambda_i \mathbf{p}_i = 0$ and $\sum \lambda_i = 0$ implies $\lambda_i = 0$ for all i. Alternately, given two affine combinations $\mathbf{x} = \sum \lambda_i \mathbf{p}_i$ and $\mathbf{y} = \sum \mu_i \mathbf{p}_i$, if $\mathbf{x} = \mathbf{y} \Leftrightarrow \lambda_i = \mu_i$, then the points $\{\mathbf{p}_i\}$ are affinely independent. The $k+1$ points are affinely independent if and only if the k vectors $\{\mathbf{p}_i - \mathbf{p}_0\}$ for $1 \leq i \leq k$ are linearly independent. Since there can exist at most n linearly independent vectors in $\mathbb{R}^n$, there can be at most $n+1$ affinely independent points.

Definition 2.5 (Convex Combination). An affine combination, $\mathbf{x} = \sum \lambda_i \mathbf{v}_i$, is a *convex combination* if $\lambda_i \geq 0$ for all i.

The set of all affine combinations of $\{\mathbf{p}_i\}$ is called its *affine hull*, and the set of all its convex combinations is called its *convex hull*. Given a set of two affinely independent (distinct) points, its affine hull is the line passing through them, whereas its convex hull is the line segment between them. For three affinely independent (noncollinear) points, the plane passing through them defines their affine hull, whereas the triangle formed by them defines their convex hull.

Definition 2.6 (Simplex). A *k-simplex*, $\mathcal{S}^k$, is the convex hull of $k+1$ affinely independent vertices, such that $\mathcal{S}^k = \{\mathbf{v}_i\}, \mathbf{v}_i \in \mathbb{R}^n,\ 0 \leq i \leq k \leq n$.

The simplices of the first few dimensions have special names—*vertex* for a 0-simplex, *edge* for a 1-simplex, *triangle* for a 2-simplex, and *tetrahedron* for a 3-simplex—and are illustrated in Figure 2.4.

Any subset of affinely independent points is also affinely independent, and, therefore, also defines a simplex. A *face* of $\mathcal{S}$ is the convex hull of a nonempty subset of $\{\mathbf{v}_i\}$, and it is *proper* if the subset is not the entire set. In particular, a simplex $\mathcal{S}^l$ is called a *l-face* of $\mathcal{S}^k$ for $l \leq k$ if $\mathcal{S}^l \subseteq \mathcal{S}^k$, and a *proper l-face* of $\mathcal{S}^k$ for $l < k$ if $\mathcal{S}^l \subset \mathcal{S}^k$. If $\mathcal{S}^l$ is a (proper) face of $\mathcal{S}^k$, then $\mathcal{S}^k$ is called a *(proper) coface* of $\mathcal{S}^l$. $\mathring{\mathcal{S}}$ denotes the interior of a simplex, and is given by removing all the proper faces from a simplex. The local neighborhood of a vertex $\mathbf{v}_j$ can be defined in terms of its *star* $\mathrm{St}(\mathbf{v}_j)$, which consists of all cofaces of $\mathbf{v}_j$. The star is not closed under taking faces. A proper $k-1$-face $\mathcal{S}_j^{k-1}$ of $\mathcal{S}^k$ is called its *facet* if it is constructed by removing vertex x_j from $\mathcal{S}^k$, that is, $\mathcal{S}_j^{k-1} = \{\mathbf{v}_i\}, \mathbf{v}_i \subset \mathcal{S}^k,\ 0 \leq i \leq k \leq n,\ i \neq j$.

Definition 2.7 (Simplicial Complex). A *simplicial complex*, denoted $\mathcal{M}$, is a collection of simplices such that $\mathcal{S}_i \in \mathcal{M}$ implies that all the faces of $\mathcal{S}_i$ are in $\mathcal{M}$, and the intersection of any two simplices is a face of both or empty.

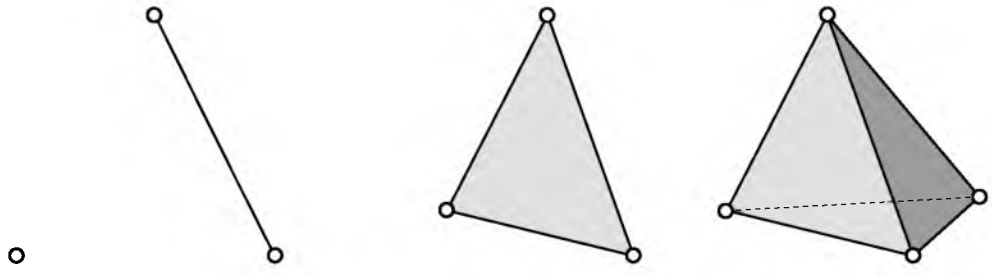


Figure 2.4. A k-simplex for $k = 0, 1, 2$, and 3 is a vertex, an edge, a triangle, and a tetrahedron, respectively.

2.4.2 Sampled Vector Fields

In contrast to smooth vector fields defined on smooth domains, experiments and simulations create sampled vector field defined on discretized domains, often simplicial complexes. This section defines sampled vector fields, and makes some assumptions to avoid degenerate cases that are hard to manage in geometric algorithms.

Definition 2.8 (Sampled Vector Field)**.** A sampled vector field $\vec{V} : \mathcal{M} \to \mathbb{R}^n$ is constructed using $m+1$ samples $\{\vec{V}_0, \ldots, \vec{V}_m\}$, $\vec{V}_i \in \mathbb{R}^n$, defined on the vertices $\{\mathbf{v}_0, \ldots, \mathbf{v}_m\}$, $\mathbf{v}_i \in \mathbb{R}^n$ of a simplicial complex $\mathcal{M}$ or a regular grid $\mathcal{G}$.

Given samples of a vector field, the field on the entire domain must be completed using interpolation. One of the most common ways is to interpolate the samples linearly within each simplex, to construct a piecewise linear vector field.

Definition 2.9 (Piecewise Linear (PL) Vector Field)**.** A piecewise linear (PL) vector field $\vec{V}$ on $\mathcal{M}$ is defined using linear interpolation of vector samples on the vertices $\{\mathbf{v}_i\}$ of $\mathcal{M}$ within each simplex.

Consider a triangle Δ_j defined by its three vertices $\{\mathbf{p}_{j_0}, \mathbf{p}_{j_1}, \mathbf{p}_{j_2}\}$. Any point $\mathbf{x}$ in Δ_j lies in the convex hull of $\{\mathbf{p}_{j_0}, \mathbf{p}_{j_1}, \mathbf{p}_{j_2}\}$, and, therefore, can be represented as a linear combination, $\mathbf{x} = \lambda_0 \mathbf{p}_{j_0} + \lambda_1 \mathbf{p}_{j_1} + \lambda_2 \mathbf{p}_{j_2}$, where λ_i are such that $0 \leq \lambda_i \leq 1$ and $\sum_{i=0}^{2} \lambda_i = 1$. The weights λ_i are called the *barycentric coordinates* of $\mathbf{x}$ with respect to the points $\{\mathbf{p}_{j_0}, \mathbf{p}_{j_1}, \mathbf{p}_{j_2}\}$. Then, given the triangle Δ_j defined by $\{\mathbf{p}_{j_0}, \mathbf{p}_{j_1}, \mathbf{p}_{j_2}\}$, and the corresponding vector samples $\{\vec{V}_{j_0}, \vec{V}_{j_1}, \vec{V}_{j_2}\}$, the field for any point $\mathbf{x}$ in Δ_j can be computed using linear (barycentric) interpolation, that is,

$$\vec{V}(\mathbf{x}) = \lambda_0 \vec{V}_{j_0} + \lambda_1 \vec{V}_{j_1} + \lambda_2 \vec{V}_{j_2}. \tag{2.11}$$

To simplify the study of PL vector fields, an important simplifying assumption is made on its samples.

Definition 2.10 (Generic Sampled Vector Field)**.** A sampled vector field defined by Definition 2.8 is called *generic*, if none of the vectors sampled at the vertices is zero, that is, $||\vec{V}_i|| > 0$, for all $i \in \{0, \ldots, m\}$.

Assuming generic sampled vector fields simplifies their analysis substantially. However, not every vector field sampled from observations or simulations is generic, since the sampled vector magnitudes can be zero. In order to obtain a generic vector field, vector samples can be perturbed symbolically using the Simulation of Simplicity.

2.4.3 Simulation of Simplicity

The Simulation of Simplicity (SoS) [49] is a general-purpose programming philosophy useful for handling degeneracies in data. Computational algorithms often work with data that may contain a number of degenerate situations, the handling of which may require addressing many special cases. The SoS philosophy is built upon the fact that degeneracies are unstable, and can be removed by arbitrarily small perturbations. Thus, the SoS provides a natural way of converting potentially degenerate input data into nondegenerate information that can be used with simpler algorithms.

A very simple example of the SoS can be found in sorting an array of integers. A degenerate situation arises when two equal numbers are encountered. The programmer must handle this situation by imposing a strict ordering on numbers, for example, by assuming that the array is indexed, and ordering for equal numbers is given by their indices. Thus, it is assumed that there exists an arbitrarily small perturbation that can be applied to either of the two equal numbers to make one number smaller than the other.

Typically, geometric algorithms are numerically not robust, and various kinds of degeneracies can occur. For example, given three points in a 2D space, let the goal be to determine their orientation. Depending upon the order of the points, the orientation can be clockwise or counterclockwise, representing two generic cases. However, the three points can also be collinear, in which case, the orientation cannot be defined. Addressing this numerical degeneracy requires additional programming efforts, and dependening upon the numerical details of the algorithm, different results may be obtained. Instead, the SoS imposes a symbolic perturbation to one of the three points to disallow them from being collinear, so that the geometric algorithm can assign a well-defined orientation.

The SoS simulates a conceptual perturbation of the input data to eliminate all degeneracies. The most attractive feature of the SoS is that the perturbation is *symbolic* and is never actually computed in practice. It is assumed to be arbitrarily small enough to simulate the generic conditions. In practice, the predicate to be implemented is represented as a determinant, whose zero value is equivalent to the degenerate case. Using symbolic perturbations, the SoS ensures that the value of the determinant is always computed to be nonzero.

The SoS will be very useful in applying implementing the framework defined in this dissertation. In particular, it is used for creating a generic vector field (see Definition 2.10) for developing Edge Maps (see Chapter 6) and for a robust detection of critical points (see Section 7.2).

2.4.4 Combinatorial Representations

The traditional representation of vector fields using samples at vertices of grids or simplicial meshes, and the completion of the field inside the cells using interpolation are affected by numerical errors, and, thus, lack robustness. Computation of flow (streamlines) typically relies on numerical integration, which further adds to the amount of error incurred, causing various problems, some of which were introduced in Chapter 1.

In contrast, the research presented in this dissertation focuses on a different type of data representation that does not suffer from these numerical errors. These *discrete representations*, which are combinatorial in nature, have been successfully applied to scalar fields. In particular, the Morse theory [137, 140, 141], which was introduced for smooth domains as early as 1963, has now been extended to discrete domains. Forman has developed a discrete Morse theory [54, 55, 56, 57] that defines the features of scalar functions on cells of a discrete domain. Since its introduction, Forman's discrete Morse theory has been applied in many applications in visualization and analysis for the purpose of obtaining robustness [78, 120, 121]. For a detailed discussion on the application of discrete Morse theory for the analysis and visualization of scalar functions, the reader may refer to the Ph.D. dissertation of Gyulassy [77].

The discrete Morse theory describes a scalar function using a discrete gradient vector field, which is represented as pairs of mesh elements, for example, vertex-edge or edge-triangle pairs, as illustrated in Figure 2.5. Formally, a vector field is described as pairwise disjoint sets of pairs of simplices of the discretized domain (for example, a triangulated mesh): $\vec{V} = \{\{\alpha, \beta\} : \beta \in \partial\alpha\}$, where α and β are simplices of $\mathcal{M}$. The primary advantage of using such a representation is that topological structures can be efficiently and robustly extracted through graph traversals. The recent work of Reininghaus and Hotz [173, 174] is also inspired by this discrete combinatorial vector field, and applies it for the extraction of

Figure 2.5. Discrete representations of vector fields can be encoded as graphs, and, therefore, can produce consistent analysis. Two discrete representations are shown: (left) Piecewise constant vector field and (right) Forman's discrete vector field [57].

topological structures from vector fields.

Another discrete representation of vector fields is a *piecewise constant (PC) vector field*, which is obtained by defining its samples at the center of the cells, for example, one vector per triangle, as illustrated in Figure 2.5. In particular, a PC vector field $\vec{V}$ on $\mathcal{M}$ is defined by a function f that assigns a vector to each triangle of $\mathcal{M}$, such that the vector is parallel to the plane of Δ, but not to any of its edges. A PC vector field is one of the simplest and coarsest ways of representing vector fields. Such vector fields are discontinuous across the edges of the triangles. Using PC vector fields, Szymczak and Zhang [197] propose an algorithm to compute the Morse decomposition by representing the set of all trajectories in the field as a finite transition graph. Szymczak [198] extends this technique to produce a superset of transition graphs to reflect the error in the field.

A third approach that can be classified in the category of combinatorial representations was proposed by Chen et al. [26, 27], who create triangle-level graphs by determining how the image of a triangle is advected and deformed by the flow. These so-called entity connection graphs (ECGs) and Morse connection graphs (MCGs) lead to a robust representation of the vector field topology for vector fields on piecewise linear (PL) manifolds. The graph is computed using the underlying PL vector field, but the accuracy of this technique remains limited by the triangle resolution. However, the technique of Chen et al. enables classification of Morse sets via Conley indices [25], which provides meaningful information, albeit at a coarse granularity.

Although robust and provably consistent, discrete representations have an important disadvantage. It must be noted that, in practice, these discrete representations are created by converting the available numerical data into combinatorial form. The geometric resolution of a discrete vector field is also limited by the mesh resolution. Thus, the benefit of combinatorial robustness comes at a high cost. Since discrete representations are coarse, it is unclear how well the computed discrete forms approximate the smooth theory or the theory of PL fields.

In this context, the work of Forman is particularly significant since it creates a new theory—not just a representation—that defines the definition and properties of the resulting features. Thus, the corresponding representations are accurate as well as consistent with respect to the discrete theory, although they may be inaccurate with respect to the numerical field and the smooth theory.

The philosophy of the work presented in Part III of this work is the same. We develop a new theory of representation that defines the type of features we can extract. Using the

corresponding representations, we provide consistency guarantees. Nevertheless, since the data from simulations is available in the numerical form, the error of conversion is encoded in the representations that we develop—an important advantage over the existing techniques. These ideas will be discussed in more detail in Chapter 7.

2.5 Vector Field Analysis

This section discusses the three main classes of analysis techniques for vector fields.

2.5.1 Streamline-based Analysis

Streamlines are a fundamental construct that can represent the complete behavior of steady vector fields. As a result, some of the most common techniques to visualize vector fields, for example, glyph-based visualization, line integral convolution [20], and image-based flow visualization [208], highlight the streamline behavior of the flow. Using streamline-based approaches, it is possible to perform higher-order [122, 180, 181, 182, 205, 214, 215] and combinatorial [26, 27, 173, 174, 197, 198] analysis, design [52, 172, 199, 225], comparison [44, 201], compression [200], and simplification [206] of vector fields, etc.

2.5.1.1 Vector field topology. The foremost example of streamline-based analysis is the *vector field topology*. Topological approaches are attractive as they produce an abstract yet complete description of the global flow behavior, as illustrated in Figure 2.6. Such tehcniques provide a simpler visualization of the flow, and, therefore, form an ideal starting point for analysis as well as visualization.

Topology-based approaches for vector fields describe the global flow structure through the behavior of its streamlines. The limit sets of streamlines—critical points and closed orbits—are of particular interest, as is the *topological skeleton* of the flow. The topological skeleton of 2D vector fields was introduced to the visualization community by Helman and Hesselink [89], who compute it by segmenting the domain using the *separatrices*—streamlines traced from saddles of the field along the directions of its eigenvectors. The critical points of the field form the nodes of the skeleton, which are connected by the separatrices. These ideas were extended to surface flows [90] and 3D flows [91] by the same authors. Globus et al. [69] proposed a similar technique for 3D vector fields by using a combination of glyphs and streamlines to depict the global structure.

At the same time, streamsurface was introduced for visualizing 3D vector fields by Hultquist [96] to represent the surface spanned by an infinite set of streamlines seeded at an arbitrary curve. The streamsurfaces were shown to correspond to separation and attachment lines. Since then the streamsurface computation has been extended to more

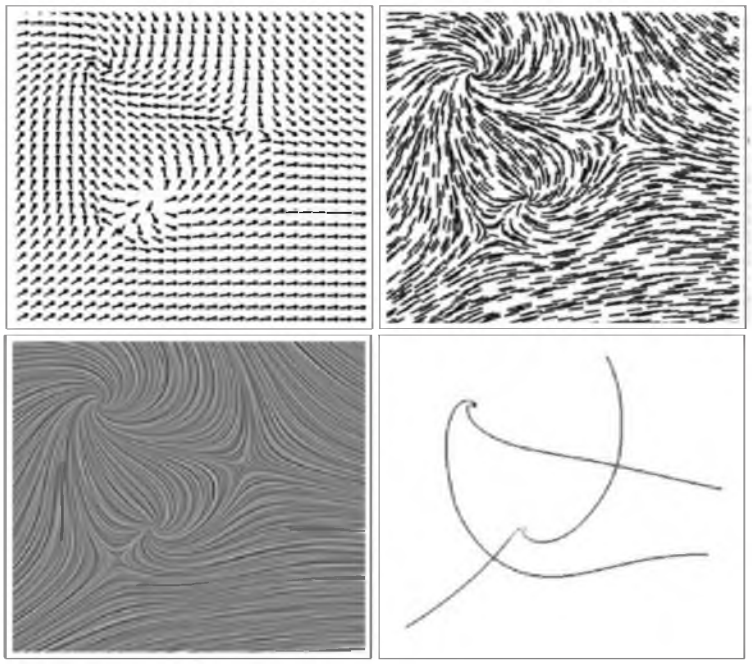


Figure 2.6. Streamline-based visualizations of steady vector fields highlight the global behavior of streamlines. As compared to glyph-based (top-left), streamline-based (top-right), and texture-based (bottom-left) visualizations of vector fields, the topology-based (bottom-right) visualization provides an abstract summary of the field by displaying only the important features. The topological skeleton of a 2D vector field is a graph with critical points as nodes connected by saddle separatrices as edges.

efficient and generalized techniques, for example, by Scheuermann et al. [179] and Garth et al. [66]. Some of the other techniques for visualizing 3D vector field topology include those proposed by Thiesel et al. [202] and Mahrous et al. [132].

The following sections discuss the three important components of topological skeleton: streamlines, critical points, and vortices. For a detailed discussion on topological methods in vector field visualization, the reader may refer to the surveys by Garth and Tricoche [65], Laramee et al. [116], Post et al. [168], and Scheuermann and Tricoche [183].

2.5.1.2 Tracing streamlines. Streamlines are typically computed using numerical integration of varying accuracy, such as the Euler integration or the Runge-Kutta (RK) techniques [170]. These techniques take a small step in the current direction of the vector, and recompute the new direction using an interpolation scheme to repeat the process. The higher-order RK methods are known to be more accurate than the Euler integration. Higher-order means greater accuracy, but more processing time. Another factor that influences the correctness and time behavior is the step-size. A shorter step-size in particle trajectory calculation means the need for more steps and a longer process. Each integration step

introduces some error, which is typically known as an upper bound with respect to the step-size. During the integration, however, these small errors get compounded, and it becomes nontrivial to determine the net error incurred during the streamline tracing.

One way to avoid numerical integration errors is to use an analytical solution to streamlines. The local exact method (LEM) of Kipfer et al. [108], which follows the lead of Nielson and Jung [149], solves an ordinary differential equation for piecewise linear vector fields defined on simplicial meshes, making it possible to represent the position of the particle as a function of time. Although the solution is more expensive than numerical integration, LEM can compute the exact path of a particle in a triangle when its entry point to the triangle is known. Consequently, it removes the need to perform step-wise numerical integration, and, hence, is free from the cumulative integration error. However, the exit point is calculated numerically as an intersection with the triangle edges, which is still prone to numeric errors. Despite this, it is the most accurate technique available since it does not incur integration error. As a result, the LEM will be used for streamline integration in Chapters 6 and 7.

2.5.1.3 Detection of critical points. Critical points are singularities (zeros) of the field, and can be classified into various kinds depending upon their behavior. Figure 2.7 shows the classification of nondegenerate singularities based on the eigenvalues of the Jacobian of the field, as explained by Helman and Hesselink [89].

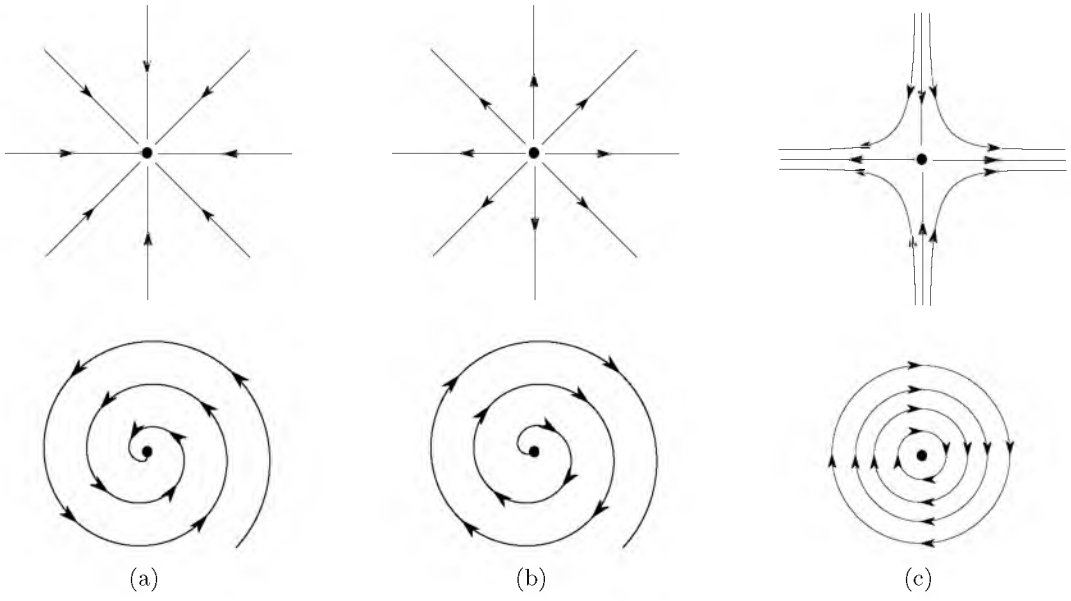


Figure 2.7. First-order critical points in a vector field can be classified based upon the eigenvalues of the Jacobian of the field computed at the critical point. (a) Sink and attracting focus; (b) source and repelling focus; and (c) saddle and center.

Most of the early attempts at identification of critical points were based on numerical analysis. For example, isolated nondegenerate singularities were identified using numerically integrated tangent curves (streamlines) [89]. Lavin et al. [118] extract singularities in a linear vector field $\vec{V} = A\mathbf{x} + \vec{o}$ by numerically solving the system $A\mathbf{x} = \vec{0}$ for each cell in a triangulated domain. This technique was extended to 3D by Batra and Hesselink [10].

The detection of singularities has also been extended to higher-order singularities. Scheuermann et al. [180, 181, 182] extract higher-order critical points by identifying regions whose winding number is greater than 1 or less than -1, and using a polynomial approximation to represent the field. Tricoche et al. [205] analyze higher-order singularities in 2D by partitioning the neighborhood of the singularity into sectors of different flow behavior. The topological analysis of higher-order singularities provides a foundation for the design and simplification of vector fields. Tricoche et al. [206] simplify the topology of vector fields by merging clustered singularities within a convex polygon into higher-order singularities. These ideas have been extended to 3D by Weinkauf et al. [214, 215]. It is more challenging to identify singularities in nonlinear vector fields. Li et al. [122] subdivide the simplicial mesh and compute the vector field by side-vertex interpolation in polar coordinates. Singularities are then ensured to be located at the vertices.

In general, detection of singularities can be reformulated as solving nonlinear systems of equations. The Newton-Raphson method and Broyden's method can be used to solve such systems. However, techniques aimed at solving generic nonlinear systems are sensitive to perturbation and not guaranteed to find all the solutions. For multivariate rational splines, Elber and Kim [50] apply the bisection method to localize the potential regions containing roots. However, computational complexity is a major concern with their method. In contrast, the proposed work aims at computing critical points consistently in both 2D and 3D in a significantly more efficient manner.

2.5.2 Pathline-based Analysis

Pathlines capture the behavior of particles advected in the flow (vector field); they provide a Lagrangian description of the flow. As a result, analysis based on pathlines provides insights about the spatiotemporal behavior of the flow, and is the most fundamental type of analysis. Pathlines are typically traced through numerical schemes, such as Runge-Kutta (RK) integration, and are typically parallelized on particles. Theisel et al. [203] use pathlines to analyze and visualize unsteady vector fields by defining pathline-oriented topology.

2.5.2.1 Finite-time Lyapunov exponent. The Lyapunov exponents are defined to characterize the rate of separation of infinitesimally close trajectories as time approaches infinity [147]. The idea behind the finite-time Lyapunov exponent (FTLE) is to apply this concept for a finite amount of time. The FTLE measures the coherence of the flow in terms of the trajectories of closely seeded particles considering the finite amount of time. The concept of the FTLE as a means to describe and analyze coherent features in unsteady flows was introduced by Haller [80]. Haller presents FTLE as a geometric approach, aimed at characterizing coherent structures in terms of preservation of certain stability types of the velocity gradient along the path of a particle. This initial research generated a significant interest in FTLE and its applications to the structural analysis of transient flows. Garth et al. focus on accelerated computation of FTLE fields and their visualization possibilities in both 2D [64] and 3D [63]; Sadlo and Peikert compare Lagrangian approaches to vector field topology [176], and propose a method for the adaptive computation of coherent structures and examine several variants of FTLE [175]. Another variant of Lyapunov exponent called the finite-size Lyapunov exponent (FSLE) exists; the FSLE measures the time taken by particles to get separated by a finite amount. A comparative study of the FTLE and the FSLE was provided by Peikert et al. [160].

2.5.2.2 Lagrangian coherent structures. The Lagrangian coherent structures (LCSs) play an important role in the analysis of unsteady flows. The idea of coherent structures was developed about 50 years ago in the context of semi- or fully turbulent flows [98]. One of the first observations of such large-scale structures was made by Roshko and Brown [18] regarding a turbulent plane mixing layer using shadowgraphs. Haller [81] and Shadden et al. [187] showed that the LCS can be represented as the ridges of the FTLE field. The LCS forms a skeleton of the unsteady flows in terms of so-called hyperbolic material lines that dominate the overall flow structure. Hence, the LCS and the FTLE have become integral components of the analysis and visualization of unsteady flows. However, Haller [83] recently showed that detecting the LCS as ridges of the FTLE field is not accurate, since this approach may create false positives and false negatives.

2.5.3 Indicator-based Analysis

Since the direct investigation of vector fields is very challenging, an often-used alternative is to perform an indirect analysis through scalar indicators derived from the flow, which may represent particle-based (Lagrangian) or point-based (Eulerian) properties of the flow. The simplest example of this class of techniques is analyzing the velocity magnitudes, instead of actual velocity vectors.

These indicator-based approaches have a distinct advantage over other types of vector field analysis, since in general, analysis of scalar fields is much more advanced than that of vector fields, and, therefore, indicators can be analyzed using simpler and more robust techniques. Furthermore, whereas both streamlines and pathlines depend upon the assumed reference frame, some commonly used and carefully chosen indicators are Galilean invariant, meaning they are invariant to uniform motion—a particularly useful property for analysis (see Section 2.3.6.1).

The most prominent example of indicator-based analysis is that in the case of vortex detection. Since there does not exist any universally accepted definition of vortices, they are typically extracted using one of the many available indicators: Q-[97], Δ-[30], or λ_2-[100] criteria. These indicators capture different properties of the flow, which are believed to represent rotational behavior or vorticity. Once these indicators have been computed, vortex regions are typically identified as regions bounded by their isosurfaces. Since these indicators are defined using the velocity gradient tensor, they are Galilean invariant—yet another advantage of detecting vortices using these indicators. The use of the finite-time Lyapunov exponent (FTLE) [80] for the detection of material boundaries is another example of such techniques, where the FTLE is a Galilean-indicator scalar indicator that captures particle separation behavior in the flow, and the ridges of the FTLE are used as a proxy for material boundaries.

Although such indicator-based techniques are very common, they have some serious disadvantages. First of all, as the name suggests, they are only "indicators," and may not actually correspond to the feature of interest. For example, Q-criterion measures the excess of the rate of rotation over the rate of strain in all directions; the λ_2-criterion looks for this excess only on a specific plane [24, 97, 100], whereas the true behavior of vortices may not correspond to either. In the case of the FTLE, it is believed that their ridges represent material boundaries, thus justifying the use of the FTLE as an indicator. However, this belief was disproved recently, and it was shown that particles may cross over the ridges of the FTLE [83].

Furthermore, by definition, the indicators are designed to capture individual properties of the flow. They must discard all other information, and cannot provide a complete analysis of flows. For example, using indicators for vortex detection disregards the behavior of the flow away from the "vortical regions" detected by these indicators, and the FTLE loses all information about the direction of the flow. Finally, these techniques typically require thresholds, for example, above or below which a region is considered to be vortical.

Depending upon the choice of this highly data-dependent parameter, the analysis can produce false positives and false negatives, as illustrated in Figure 2.8.

Nevertheless, despite the inherent limitations of indicator-based techniques, they are often used, especially for the detection of vortices and material boundaries, because of the lack of better alternatives.

2.6 Uncertainty Visualization

The quantification and visualization of uncertainty was identified by Johnson as one of the top research problems in scientific visualization [101, 102]. The fundamental goal here is to understand the errors incurred in the analysis and visualization of scientific data, and show them to the user in order to provide an honest visualization of the data. For example, instead of showing an isosurface representing the tumor cells in an MRI scan of the brain, visualizing the envelope representing the error bars of the isosurface is more useful for drawing scientific insights from the results. The Ph.D. dissertation of Potter [169] discusses uncertainty visualization in great detail. She identifies four main types of uncertainty: (a) experimental, (b) geometric, (c) simulation, and (d) visualization. She proposes a new framework for visualizing ensembles of data as uncertainty visualization. Interested readers may also visit her webpage (`http://www.sci.utah.edu/~kpotter/Library/Catalogs/uncertaintyVis/`), where she maintains a library of scientific publications on this topic. A recent state-of-the-art article [13] also summarizes the advances in uncertainty visualization.

Among the first to address the challenges of uncertainty visualization in vector fields were Pang et al. [156], who identify three sources of uncertainty: (a) uncertainty in data, (b) uncertainty due to derivation, and (c) uncertainty due to visualization. The earlier work

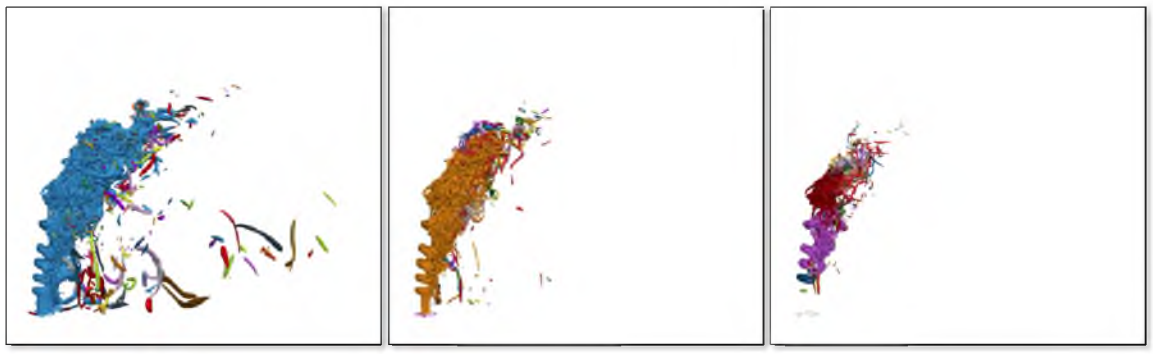


Figure 2.8. Vortex detection using indicators requires thresholds, which, in the absence of prior knowledge, can create false positives and false negatives. Visualization of three thresholds of the λ_2 indicator is shown for the jet in cross-flow. It is unclear which one of these, if any, represents the "correct" vortices.

on uncertainty visualization in vector fields concentrated on developing extended glyphs for vector samples to represent uncertainty [156, 221], by representing errors in angle or magnitudes or both, using shapes and orientations of glyphs, or varying visual channels such as hue, saturation, and transparency. Uncertainty can also be captured using the flow radar glyphs [94], which have been proposed to capture the temporal dimension in unsteady flows, or through the animation of glyphs [226].

Other efforts focused on comparing streamlines and pathlines that were computed using different integration schemes, step-sizes, and data resolutions [124, 209]. Such techniques typically use tubes, ribbons, or similar structures to visualize streamlines or pathlines, where color, thickness, or transparency of the tube is used to represent uncertainty in the location along the corresponding lines.

Recently, Otto et al. describe a method to obtain uncertain topological segmentations by sampling in a random 2D vector field [153], and later extending it to 3D vector fields [154]. This method tries to simulate the uncertainty in data by defining it using probability density functions. Their technique disregards the uncertainty due to the computation and visualization itself. Compared to their approach, the work presented in this dissertation assumes the given data as certain, and enables visualization of errors introduced during computation and visualization only.

2.7 Vector Field Decompositions

In order to enable streamline-based analysis for unsteady flows, Part III of this dissertation decomposes a given flow into appropriate components. Therefore, it is important to understand the following two important decompositions.

2.7.1 The Helmholtz-Hodge Decomposition (HHD)

The Helmholtz-Hodge decomposition [210] is one of the fundamental theorems in fluid dynamics. This decomposition describes a vector field in terms of its incompressible and irrotational components, thus simplifying any subsequent analysis as important properties, such as incompressibility and vorticity, can be studied on the components directly. Following its success and popularity, numerous interpretations and variants have been proposed that are useful in a large number of applications, as illustrated by Figure 2.9. This section provides a brief discussion of the HHD and its properties, with a particular focus on its uniqueness and boundary conditions.

As described by von Helmholtz in his seminal paper [210, Section 1], the motion of a volume element of a continuous fluid media in $\mathbb{R}^3$ consists of

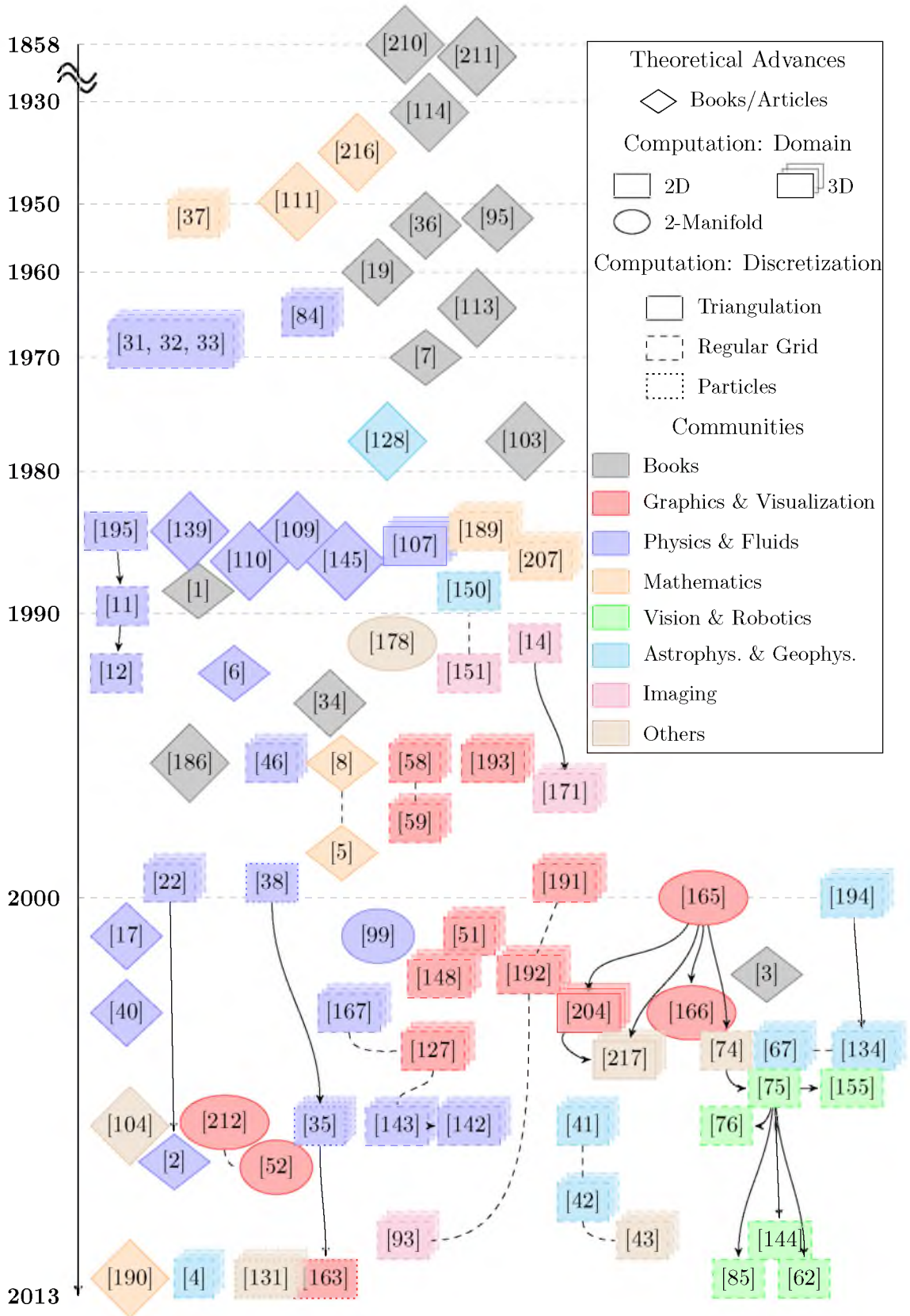


Figure 2.9. Chronological chart showing the development of the theory and computation of the HHD. Solid arrows and dashed lines represent direct extension and a weaker connection of articles, respectively.

- expansion or contraction in three orthogonal directions,
- rotation about an instantaneous axis, and
- translation.

Expansion/contraction can be represented as the gradient of a scalar potential function since it is irrotational. Similarly, rotation can be represented as the curl of a vector potential function since it is incompressible. Translation, however, being both incompressible and irrotational, can be represented as either the gradient of a scalar potential or the curl of a vector potential. Equivalently, translation can also be represented as a separate harmonic component. Depending upon the domain, and how the translation is represented, the HHD can take different forms.

2.7.1.1 Two-component HHD in infinite space. Given a smooth vector field $\vec{V} : \mathbb{R}^n \to \mathbb{R}^n$, for $n = 2, 3$, with suitable asymptotic behavior at infinity, that is, $\vec{V} \to \vec{0}$ as $\mathbf{x} \to \infty$, the vector field can be decomposed into its curl-free (irrotational) and divergence-free (incompressible) components by defining them as the gradient of a scalar potential and the curl of a vector potential, respectively, that is,

$$\begin{aligned} \vec{V} &= \vec{d} + \vec{r}, \\ &= \nabla D + \nabla \times \vec{R}, \end{aligned} \tag{2.12}$$

where by definition, $\vec{d} = \nabla D$ is irrotational ($\nabla \times \vec{d} = \vec{0}$), and $\vec{r} = \nabla \times \vec{R}$ is incompressible ($\nabla \cdot \vec{r} = 0$). These incompressible and irrotational properties lead to the following equalities:

$$\begin{aligned} \nabla \cdot \vec{d} &= \nabla \cdot \vec{V}, \\ \nabla \times \vec{r} &= \nabla \times \vec{V}. \end{aligned} \tag{2.13}$$

Substituting $\vec{d} = \nabla D$ and $\vec{r} = \nabla \times \vec{R}$ in (2.13), two Poisson equations are obtained.

$$\begin{aligned} \Delta D &= \nabla \cdot \vec{V}, \\ \vec{\Delta} \vec{R} &= -\nabla \times \vec{V}. \end{aligned} \tag{2.14}$$

where Δ is the (scalar) Laplacian, that is, $\Delta = \nabla^2$, and $\vec{\Delta}$ is the vector Laplacian, that is, $\vec{\Delta}\vec{R} = \nabla(\nabla \cdot \vec{R}) - \nabla \times (\nabla \times \vec{R})$. The Expressions (2.14) also utilize the fact that $\nabla \cdot \vec{r} = 0$ implies $\nabla \cdot \vec{R} = 0$ due to the gauge condition [68]. Under the vanishing condition at infinity, the two potentials are computed using the integral solution (similar to Equation (2.5)) as

$$\begin{aligned} D(\mathbf{x}_0) &= \int_{\mathbb{R}^n} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\nabla \cdot \vec{V}(\mathbf{x}) \right) \mathrm{d}\mathbf{x}, \\ \vec{R}(\mathbf{x}_0) &= -\int_{\mathbb{R}^n} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\nabla \times \vec{V}(\mathbf{x}) \right) \mathrm{d}\mathbf{x}, \end{aligned} \tag{2.15}$$

where $\mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)$ is the free-space Green's function, discussed in Section 2.1. Due to the vanishing condition at infinity, the integrals in (2.15) are guaranteed to converge. It must

be noted that the potentials D and $\vec{R}$ are unique up to a constant; hence, the decomposition is unique.

2.7.1.2 Two-component HHD on simply-connected finite domains. The HHD is defined uniquely for vanishing vector fields on infinite domains $\mathbb{R}^n$, since no harmonic flow can exist on infinite domains, and it is possible to decompose the incompressible and irrotational components. Equivalently, it can be said that for infinite domains, the potentials D and $\vec{R}$ can be computed using Expressions (2.15). However, on finite domains with boundary, a harmonic flow can exist. It is known from the potential theory that for a simply-connected domain, a harmonic function is completely determined using the boundary values. Therefore, in such cases, the potentials can be computed with additional boundary integral terms.

Given a smooth vector field $\vec{V} : \Omega \to \mathbb{R}^n$, where $\Omega \subset \mathbb{R}^n$ for $n = 2, 3$, the corresponding potentials D and $\vec{R}$ are defined as

$$\begin{aligned} D(\mathbf{x}_0) &= \int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\nabla \cdot \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x} - \oint_{\partial\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\vec{n} \cdot \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x}, \\ \vec{R}(\mathbf{x}_0) &= -\int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\nabla \times \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x} + \oint_{\partial\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\vec{n} \times \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x}, \end{aligned} \tag{2.16}$$

where $\vec{n}$ is the outward normal to the boundary. Here, the components $\vec{d}$ and $\vec{r}$ are often called the *longitudinal* and the *transverse* components [7, 196]. It is important to note that this form of decomposition is defined only for simply-connected subsets of $\mathbb{R}^n$. A detailed derivation of Expressions (2.16) is provided in Appendix A.1.4.

2.7.1.3 Two-component HHD on finite domains. For simply-connected subsets of $\mathbb{R}^n$, Expressions (2.16) can be used to compute the potentials uniquely. However, in the case of general (possibly nonsimply connected) finite domains, a different way of computing the HHD is required. In such cases, either the irrotational or the incompressible component is typically computed by solving either $\vec{d} = \nabla D$ or $\vec{r} = \nabla \times \vec{R}$, respectively, by imposing some boundary condition, and the second component is computed as the remainder. Some of the more common boundary conditions are discussed below.

- Chorin and Marsden [34] state that uniqueness can be obtained by requiring the incompressible component to be tangential to the boundary. Thus, the HHD is defined by the following:

$$\begin{aligned} \vec{\Delta}\vec{R} &= -\nabla \times \vec{V} && \text{on } \Omega, \\ \vec{n} \cdot (\nabla \times \vec{R}) &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{2.17}$$

  where $\vec{n}$ is the exterior normal to the boundary.

- Denaro [40] states that the boundary conditions can also be imposed on the irrotational component. In particular, he states that the irrotational component can be made normal to the boundary as follows:

$$\begin{aligned} \Delta D &= \nabla\cdot\vec{V} && \text{on } \Omega, \\ \vec{n}\times\nabla D &= \vec{0} && \text{on } \partial\Omega, \end{aligned} \tag{2.18}$$

- If the physical system guiding the decomposition specifies the tangential component of the flow, $\vec{V}_t = \vec{n}\times\vec{V}$, a unique HHD can be computed by solving the following Poisson system.

$$\begin{aligned} \vec{\Delta}\vec{R} &= -\nabla\times\vec{V} && \text{on } \Omega, \\ \vec{n}\times(\nabla\times\vec{R}) &= \vec{n}\times\vec{V} && \text{on } \partial\Omega. \end{aligned} \tag{2.19}$$

- Instead, if the physical system guiding the decomposition specifies the normal component of the flow, $V_n = \vec{n}\cdot\vec{V}$, the following system can be used to compute a unique HHD.

$$\begin{aligned} \Delta D &= \nabla\cdot\vec{V} && \text{on } \Omega, \\ \vec{n}\cdot\nabla D &= \vec{n}\cdot\vec{V} && \text{on } \partial\Omega. \end{aligned} \tag{2.20}$$

In addition to uniqueness, these boundary conditions also satisfy *orthogonality* with respect to the L_2-norm—a property important for fluid simulations. The proofs for the existence, uniqueness, and orthogonality for these decompositions are given in Appendix A.1. It can be verified that, in the two-component form of the HHD, the boundary conditions (2.17) and (2.20) produce the same decomposition due to the completeness property, that is, $\vec{n}\cdot(\nabla\times\vec{R}) + \vec{n}\cdot\nabla D = \vec{n}\cdot\vec{V}$. Similarly, the boundary conditions (2.18) and (2.19) produce the same decomposition. Next, the three-component form of the HHD is discussed, for which the equivalence of these pairs of boundary conditions does not hold, unless the harmonic component is zero.

2.7.1.4 Three-component HHD on finite domains. Another way of expressing the HHD on finite domains is to represent the harmonic flow as a separate component, leading to the three-component form of the HHD, as illustrated in Figure 2.10.

$$\vec{V} \;=\; \vec{d} + \vec{r} + \vec{h}, \tag{2.21}$$

where $\vec{d}$ and $\vec{r}$ have the same meaning as in the two-component form, and $\vec{h}$ is a harmonic flow, that is, $\nabla\cdot\vec{h} = 0$ and $\nabla\times\vec{h} = \vec{0}$. This is, by far, the most common formulation of the HHD, and is used extensively in flow analysis and visualization. This formulation requires two boundary conditions imposed on each of the potentials, leading to the uniqueness of

the decomposition. The most common boundary conditions are a combination of (2.18) and (2.17). For brevity, this dissertation refers to these boundary conditions as the *NP (normal-parallel)* boundary conditions. The HHD obtained using the NP boundary conditions is denoted as HHD_{NP}.

Definition 2.11 (The HHD with NP boundary conditions)**.** The HHD with NP boundary conditions, denoted as HHD_{NP}, is a three-component form of the HHD, that is,

$$\text{HHD}_{\text{NP}} = \vec{d}_{\text{NP}} + \vec{r}_{\text{NP}} + \vec{h}_{\text{NP}},$$

computed using the NP boundary conditions, that is, its components satisfy the following:

- $\vec{n} \times \vec{d}_{\text{NP}} = \vec{0}$, that is, the irrotational component is normal to the boundary.
- $\vec{n} \cdot \vec{r}_{\text{NP}} = 0$, that is, the incompressible component is parallel to the boundary.

where $\vec{n}$ is the outward normal to the boundary.

2.7.1.5 Simplified expressions for the HHD in 2D. In $\mathbb{R}^2$ (or a 2-manifold embedded in $\mathbb{R}^3$), curl is a scalar quantity in the upward normal direction. Hence, instead of needing a vector potential $\vec{R}$, the incompressible component can be represented using the curl of a scalar potential R to simplify the decomposition as

$$\begin{aligned} \vec{V} &= \vec{d} + \vec{r} + \vec{h}, \\ &= \nabla D + \mathcal{J}\nabla R + \vec{h}, \end{aligned} \tag{2.22}$$

and the Poisson equations (2.14) can be reduced to the following:

$$\begin{aligned} \Delta D &= \nabla \cdot \vec{V}, \\ \Delta R &= -\nabla \cdot \mathcal{J}\vec{V}, \end{aligned} \tag{2.23}$$

where R is a scalar potential, and the operator $\mathcal{J}$ rotates a 2D vector counterclockwise by $\pi/2$. Equations (2.22) and (2.23) have been derived in Appendix A.1.5.

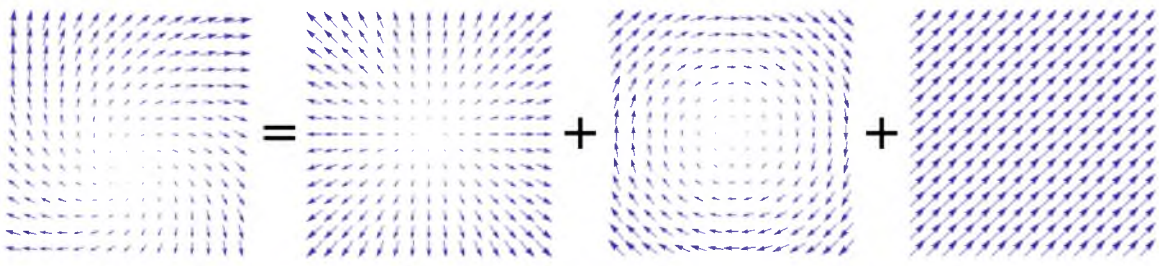


Figure 2.10. The HHD decomposes a vector field into three components: an irrotational, an incompressible, and a harmonic component. The figure shows the three-component form of the HHD.

2.7.1.6 Boundary conditions of the HHD. The most important property of the NP boundary conditions is that the resulting decomposition is L_2-orthogonal. However, the NP boundary conditions are not necessary for orthogonality. There may exist other boundary conditions, such as suitable periodic boundaries conditions, that can be applied to enforce orthogonality. For specific applications, the boundary conditions can be simplified while maintaining orthogonality. For example, considering no-flow ($V_n = 0$ on $\partial\Omega$) or no-slip $\left(\vec{V} = 0 \text{ on } \partial\Omega\right)$ boundary conditions imposed on the fluid, the boundary condition (2.18) translates into $\vec{n} \cdot \nabla D = 0 = \frac{\partial D}{\partial \vec{n}}$, which still guarantees orthogonality in the decomposition. Similarly, for a no-slip boundary, $\vec{n} \times \vec{r} = 0$ maintains orthogonality. If orthogonality is not required, the HHD can take various forms depending upon the boundary conditions. More general and nonorthogonal HHD-like decompositions, $\vec{V} = \nabla D + \nabla \times \vec{R}$, and their existence and uniqueness properties under different boundary conditions on D or $\vec{R}$ have been studied [5, 8].

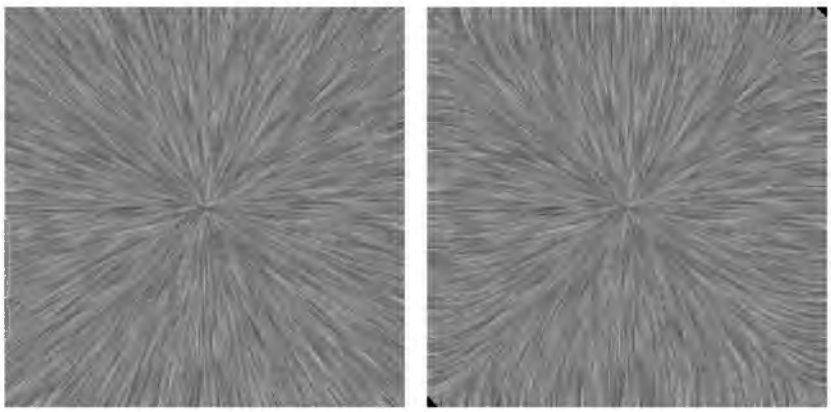
For simply-connected domains with boundary, a harmonic component is obtained as the residual when boundary conditions are used to compute the other two components. This harmonic component can be represented as the gradient of a scalar function, or the curl of a vector function. However, for more general (nonsimply-connected) domains, this harmonic component $\vec{h}$ in the HHD can be further split by applying boundary conditions:

$$\begin{aligned} \vec{h} &= \nabla H_s + \vec{h}_1 \qquad \left(\vec{n} \cdot \vec{h}_1 = 0 \text{ at } \partial\Omega\right), \\ \vec{h} &= \nabla \times \vec{H}_v + \vec{h}_2 \qquad \left(\vec{n} \times \vec{h}_2 = 0 \text{ at } \partial\Omega\right). \end{aligned} \tag{2.24}$$

Such a decomposition is often referred to as the *Hodge-Morrey-Friedrichs decomposition* (HMFD). Note that the HMFD is a general form of the HHD, and is always valid, since for simply-connected domains, $\vec{h}_1 = \vec{h}_2 = 0$, the HMFD reduces to HHD. For more details on general forms of HHD, the reader may refer to Schwarz [186].

2.7.1.7 Choice of boundary conditions in flow analysis and visualization. In topological analysis and visualization of flows, orthogonality might be desirable for obtaining a unique decomposition. Assuming no boundary conditions superimposed by the fluid model, the treatment of the issue is typically less involved. Polthier and Preuß [165, 166] as well as Tong et al. [204] enforce the NP boundary conditions for the extraction of all three components of the HHD.

In general, applications in the analysis and visualization community follow the literature from the fluids community, and typically use the NP boundary conditions. Nevertheless, if these boundary conditions are incompatible with the underlying flow, they may create substantial artifacts. For example, consider a purely irrotational flow as shown in Figure 2.11.



(a) (b)

Figure 2.11. The use of boundary conditions can cause artifacts in the HHD if the original flow does not match the conditions. (a) The original vector field $\left(\vec{V}(x,y) = (x,y)\right)$. (b) The irrotational component $(\vec{d})$ recovered after HHD. Artifacts are seen near the boundary since the NP boundary conditions force the flow to be normal to the boundary. The incompressible component $\vec{r}$ is zero for this example, and the harmonic component $\vec{h}$ also shows similar artifacts due to residual error.

In this case, the flow is not normal everywhere on the boundary. If the HHD is computed for this vector field, the incompressible component will be zero. However, in light of boundary conditions $\vec{n} \times \vec{d} = 0$, it can be easily observed that at the boundary, a disagreement occurs between the original vector field and the boundary conditions. The manifestations of such disagreements are stronger when the critical point lies closer to the boundary. Thus, these artifacts are the by-product of the boundary conditions, and are admitted into the solution in order to obtain a unique and orthogonal decomposition. These artifacts have been acknowledged by many researchers [46, 130, 217], and are particularly important in the context of this work, as will be discussed later in Chapter 3.

Guo et al. [74, 75] and others, extending the ideas of Polthier and Preuß (see Figure 2.9), do not specify any boundary conditions. According to Guo et al., setting the potentials as zero at only one vertex is sufficient to compute the decomposition uniquely, although they do not make any claims about the orthogonality of the decomposition. For better accuracy in results, they propose to further decompose the harmonic component recursively. The need for a recursive solution remains unclear, since with consistent boundary conditions, there should exist a unique solution up to numerical precision that can be computed directly as in the methods of Polthier, Tong, and others. In the case of Guo et al., this error seems to be due to the lack of proper boundary conditions, since the correct boundary conditions must be specified to obtain a unique decomposition.

Recently, Petronetto et al. [163] have computed the 2D HHD with a different set of boundary conditions, which are opposite to the NP boundary conditions. According to these authors, these boundary conditions are the usual boundary conditions for the decomposition. For shorthand, these boundary conditions will be referred to as the *PN (parallel - normal)* boundary conditions. According to the PN boundary conditions,

- $\vec{n} \cdot \vec{d} = 0$, that is, the irrotational component is parallel to the boundary.
- $\vec{n} \times \vec{r} = \vec{0}$, that is, the incompressible component is normal to the boundary.

Petronetto et al. draw inspiration from Colin et al. [35], who compute the null-divergence flow from a given flow using the projection. Colin et al. use the Neumann boundary conditions $\frac{\partial D}{\partial \vec{n}} = 0$ for boundaries that are neither no-flow nor no-slip. As has been discussed, $\frac{\partial D}{\partial \vec{n}} = 0$ satisfies orthogonality only when a no-slip or no-flow boundary is in place, that is, $V_n = 0$. Appendix A.1.6 shows that the PN boundary conditions are, in general, invalid, and cannot be used to compute the HHD of a general vector field.

2.7.1.8 Computation using least-squares finite-element method (LSFEM). Many of the recent techniques to compute the discrete HHD employ a least square technique to compute the potential functions corresponding to the incompressible and irrotational components. These techniques work for a finite-element framework with piecewise constant (PC) vector fields (defined on the triangles of the mesh), resulting in piecewise linear (PL) potential functions (defined on the vertices of the mesh).

Polthier and Preuß [166] provide definitions for discrete divergence and discrete rotation on a 2-manifold similar to weak derivatives in finite-element methods, and use them to propose a simple implementation of the discrete HHD. The 2D HHD given by Expressions (2.22) is computed by solving the Poisson equations (2.23) using a variational approach to minimize energy functions for the irrotational and incompressible components. Finally, the harmonic component is computed as the residual, and the error caused by the approximation or the boundary conditions is reflected in this component.

The discrete divergence and discrete rotation for a PC vector field are defined as PL scalar functions. In the following, let St($\mathbf{v}$) denote the star of the vertex $\mathbf{v}$.

- Discrete Divergence ($\nabla \cdot \vec{V}$): For a vertex $\mathbf{v}$, the discrete divergence can be defined as the net flow normal to the boundary of St($\mathbf{v}$). It can be calculated as the integrated dot product of the vector with the outward normal for every triangle in St($\mathbf{v}$). The

outward normal can also be calculated by rotating the normalized edge vector $\vec{c}$ (of the triangles in St(**v**)), which does not have **v** as one of its vertices.

$$\nabla\cdot\vec{V}(\mathbf{v}) = \sum_{i=1}^{k} \left\langle \vec{V}_i, \nabla\varphi_{\mathbf{v}_i} \right\rangle = -\frac{1}{2}\sum_{i=1}^{k} \left\langle \vec{V}_i, \mathcal{J}\vec{c}_{\mathbf{v}_i} \right\rangle.$$

where k is the number of triangles in St(**v**), $\nabla\varphi_{\mathbf{v}}$ is the Lagrange basis function corresponding to vertex **v**, and $\mathcal{J}$ is the counterclockwise rotation of a vector.

- Discrete Rotation ($\nabla\times\vec{V}$): For a vertex **v**, the discrete rotation can be defined as the net flow tangential to the boundary of St(**v**). It can be calculated as the integrated dot product of the vector with the rotated outward normal for every triangle in St(**v**), where the latter is the same as the normalized edge vector $\vec{c}$ in St(**v**).

$$\nabla\times\vec{V}(\mathbf{v}) = -\sum_{i=1}^{k} \left\langle \mathcal{J}\vec{V}_i, \nabla\varphi_{pi} \right\rangle = \frac{1}{2}\sum_{i=1}^{k} \left\langle \vec{V}_i, \vec{c}_{\mathbf{v}_i} \right\rangle.$$

Using the above definitions, the following two energy functionals are minimized:

$$\begin{aligned} F(D) &= \int \left(\vec{V} - \nabla D\right)^2 = \int \left(|\nabla D|^2 - 2\left\langle \nabla D, \vec{V} \right\rangle\right) + \int |\vec{V}|^2, \\ G(R) &= \int \left(\vec{V} - \mathcal{J}\nabla R\right)^2 = \int \left(|\mathcal{J}\nabla R|^2 - 2\left\langle \mathcal{J}\nabla R, \vec{V} \right\rangle\right) + \int |\vec{V}|^2. \end{aligned}$$

To achieve these minimizations, the derivatives of both the functionals should vanish at every vertex. Explicit representations of the derivatives at p are given by

$$\begin{aligned} \frac{\mathrm{d}F}{\mathrm{d}D_{\mathbf{v}}} &= -\sum_{i=1}^{k} \left\langle \left(\nabla D_i - \vec{V}_i\right), \mathcal{J}\vec{c}_{\mathbf{v}_i} \right\rangle = 0, \\ \frac{\mathrm{d}G}{\mathrm{d}R_{\mathbf{v}}} &= \sum_{i=1}^{k} \left\langle \left(\nabla R_i + \mathcal{J}\vec{V}_i\right), \mathcal{J}\vec{c}_{\mathbf{v}_i} \right\rangle = 0. \end{aligned} \tag{2.25}$$

Once the irrotational and incompressible components have been computed, the harmonic component is found as the residual, that is, $\vec{h} = \vec{V} - \nabla D - \mathcal{J}\nabla R$. The NP boundary conditions are used to get the unique decomposition.

2.7.2 Localized Flow Decomposition

Given a flow field $\vec{V} : \Omega \to \mathbb{R}^n$, Wiebel et al. [219, 220] represent it as the sum of a *potential* and a *localized* flow. Representing the potential flow $\vec{V}_P$ as the gradient of a scalar

function u, that is, $\vec{V}_P = \nabla u$, the following Laplace equation with a Neumann boundary condition is solved:

$$\nabla^2 u = 0 \qquad \text{on } \Omega$$
$$\vec{n} \cdot \nabla u = \vec{n} \cdot \vec{V} \qquad \text{on } \partial\Omega$$

Solving this equation gives a potential flow that matches the original field on the boundary of a subdomain, but is otherwise simple in the sense that it has vanishing divergence and curl, that is, $\nabla \cdot \vec{V}_P = 0$ and $\nabla \times \vec{V}_P = \vec{0}$. The potential flow represents the laminar flow induced by the geometry of the domain, and its boundary values, that is, $\vec{n} \cdot \vec{V}_P = \vec{n} \cdot \vec{V}$. The residual component $\vec{V}_R = \vec{V} - \vec{V}_P$ is called the localized flow (or the region-specific flow). This decomposition captures the entire divergence and curl of the original flow, and, hence, provides deeper insights into the features in the original flow. Any analysis that makes use of the these two quantities is not affected by the decomposition of the flow. All critical points (except all saddles) are represented in this component. This method works for an arbitrary subregion, with a piecewise smooth boundary.

This technique, however, has some serious drawbacks. First of all, the computation of the potential flow imposes boundary conditions, and as discussed in Section 2.1, imposing boundary conditions is equivalent to assuming unknown information about the exterior of the domain. Furthermore, the localized flow is always confined within the domain, that is, its component normal to the boundary is zero. This is a strong assumption, especially for compressible flows. For example, consider a nodal source in a closed domain, where one must expect the flow created due to the source to go across the boundary, something not possible in a localized flow. The technique also needs additional information (material density) to resolve compressible flows. Finally, as discussed by Wiebel et al. [220], the material density should be near constant for the technique to apply to unsteady flows.

PART II

LOCAL REFERENCE FRAMES

CHAPTER 3

DECOMPOSITION OF FLOW WITHOUT BOUNDARY CONDITIONS

In order to define local reference frames and to transform a given vector field into such frames, this work decomposes the given field into appropriate components. Chapter 2 discussed some decompositions for vector fields relevant for this task. In particular, an important vector field decomposition, called the Helmholtz-Hodge decomposition (HHD), was discussed. The HHD has been useful in a large number of application areas, where its typical role is to separate compressible and rotational properties of the flow. However, as such, the HHD cannot be used for the computation of local frames. The focus of the this chapter is to understand the limitations of the HHD in the context of computing reference frames. This chapter introduces a new formulation of the HHD that will be used later in Chapter 4 to compute an important type of reference frame.

The inability of the current formulations of the HHD to be useful for computing local frames is intimately connected with the use of boundary conditions in its computation. On domains with boundary, the HHD is not unique; the conventional way to compute a unique HHD is to impose boundary conditions. The use of boundary conditions in the HHD creates severe artifacts, which have already been acknowledged by many researchers [46, 130, 217]. Yet, there does not exist a good understanding of the nature of these artifacts. This chapter takes an agnostic view on the use of boundary conditions, and explores the fundamental limitations associated with them.

To address the issues of the boundary conditions, this chapter discusses a new formulation of the HHD that does not impose boundary conditions to obtain a unique decomposition. Using concepts from potential theory, a given flow is considered to be the sum of flows created due to *internal* and *external* influences with respect to a given bounded domain. In this context, the traditional way of imposing boundary conditions is equivalent to presuming to know the external influences, thus leading to a unique identification of flow due to internal influences. In contrast, the approach presented in this chapter reverses

the process by computing the flow due to the internal influences using the given data, uniquely and without any assumptions. The residual is then considered to be the flow due to external influences. As a result, the proposed decomposition is unique without prescribing boundary conditions. Consequently, no artifacts are introduced in the decomposition due to boundary conditions, and it naturally determines the component flows along the boundary in a data-driven manner. Therefore, the proposed decomposition is referred to as the *natural Helmholtz-Hodge decomposition.*

Chapter 4 will show that computing the HHD without using any boundary conditions allows the computation of meaningful local reference frames. In addition to computing local frames, the work presented in this chapter will also be useful in various applications where boundary artifacts must be avoided. In particular, many flows are generated through simulations, and the boundary conditions used during the simulation may not be known during the analysis, or may be too complex to be incorporated. Furthermore, a large number of simulations instead use an *open boundary*, where the flow at all or part of the boundary is observed as the output of the simulation, for example, in large-scale combustion simulations [60, 73, 88, 222], free surface fluid animations [188], or oceanography [119]. In such cases, a standard set of boundary conditions for the HHD may be incompatible with the data, and may lead to unphysical results.

3.1 Harmonic Flow and the Boundary of the Domain

The HHD decomposes a given flow into three components: an irrotational flow, an incompressible flow, and a harmonic flow. The fundamental reason for nonuniqueness in the HHD is that harmonic flows are both irrotational and incompressible. Therefore, one can add an arbitrary harmonic flow to any one of the three HHD components, and its negative to another, to obtain a different valid decomposition. As a result, any two valid HHDs differ only in how the harmonic flow is represented. To address nonuniqueness, it is important to first understand the nature of harmonic flows, how they are related to boundary conditions, and why imposing boundary conditions may not be a good choice.

Traditionally, uniqueness is obtained by imposing boundary conditions on the computation of the irrotational and incompressible components. For example, the most common boundary conditions require the irrotational and incompressible components to be normal and parallel to the boundary, respectively. However, from the theory of potential functions, discussed in Section 2.1, it is known that the solution to a Poisson equation computed using boundary conditions may contain a nonzero harmonic function. In the context of the

HHD, imposing boundary conditions adds certain harmonic flows to the incompressible and irrotational components to match the imposed boundary flow, and their negative to the harmonic component, which is then computed as the residual.

However, these boundary conditions are somewhat arbitrary for general flows; they may not be compatible with a given flow. For example, consider the rotational flow shown in Figure 3.1. Through the HHD of this field, it would be expected to recover the same flow as the incompressible component, with the other two components being zero. However, as the figure illustrates, the obtained incompressible component contains serious artifacts. Especially, the topology of the output field has changed with the addition of a new critical point. These artifacts are caused due to imposing a parallel flow on the boundary—a configuration incompatible with the original flow.

In general, boundary conditions create a strong coupling between the component flows and the shape and orientation of the boundary. As illustrated later in the results section (Section 3.3), when the same analytical flow is sampled on differently shaped domains, one obtains markedly different decompositions. Using these decompositions for analysis may produce significantly different results for the same flow depending on the shape of the

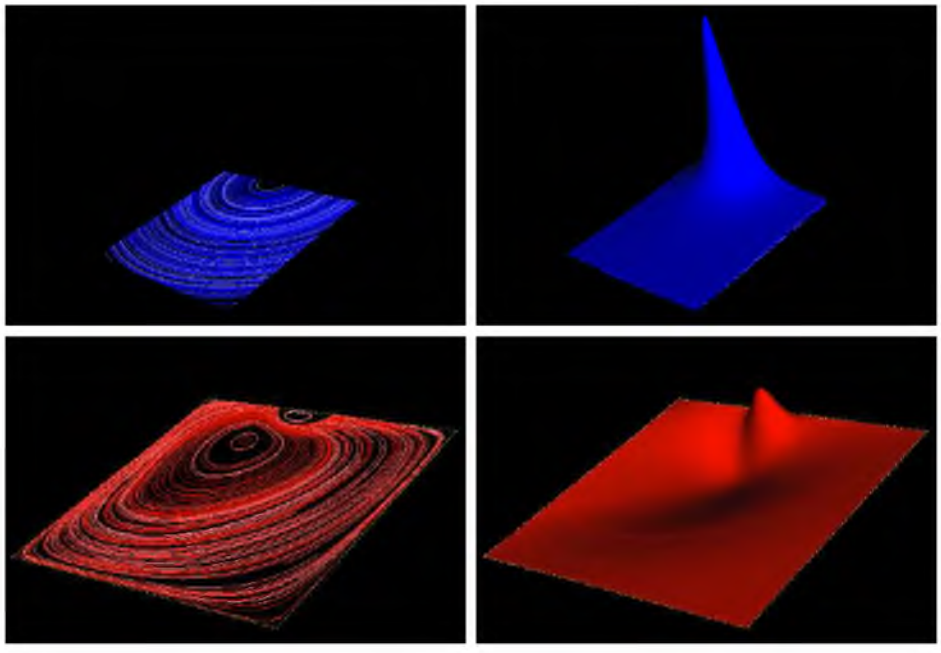


Figure 3.1. Artifacts of the boundary conditions used for computing a unique HHD—an example given by Wiebel [217]. Top: A rotational vector field (left) and the potential function used to create it (right). Bottom: The incompressible component and its potential function recovered through the HHD. Note that the rotation is inverted and an additional critical point is introduced in the output field.

domain. Given that the shape of the domain is usually determined by the region of interest and/or practical considerations such as mesh type or compute power, this dependency calls such analysis into question.

Intuitively, these examples suggest the need to identify the harmonic flow independent of the shape and orientation of the boundary. However, by definition, a harmonic function (and a harmonic flow) is determined entirely by the function values prescribed at the boundary of the domain. Equivalently, a harmonic flow is always defined with respect to the information available to (the viewpoint of) the observer. For example, as seen in Figure 3.2, considering only Ω, all of the translational flow in Ω is harmonic, and an observer would presume its HHD should contain only a harmonic component, with the other two components being zero. However, considering the domain Ω' instead, the observer notices that some of that translational flow in Ω is created and destroyed by the local divergence (in red and green) in $\Omega' \setminus \Omega$, and, therefore, is not harmonic with respect to the larger domain Ω'. As a result, the irrotational component of the HHD on Ω' should be nonzero in Ω.

It is well known that harmonic functions are associated with boundary conditions, which equivalently means that they depend upon the phenomena occurring outside the given domain. However, since external information is usually unknown, identifying harmonic flows by imposing boundary conditions on Ω is equivalent to presuming to know the flow on Ω', or rather, its influence on Ω. In contrast, it is argued here that any nonharmonic flow can be completely explained by the information available inside the domain. Therefore,

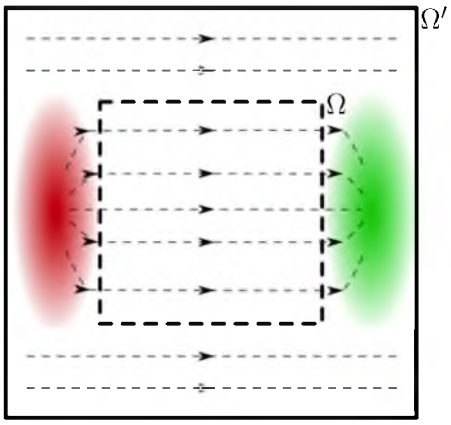


Figure 3.2. Dependence of the definition of a harmonic flow on the boundary of the domain. Seen in the context of the divergence inside Ω' (solid boundary), one associates some of the flow in Ω with the positive (red) and negative (green) divergence, and, therefore, all of it is not harmonic. However, considering Ω (dashed boundary) such that the divergence inside the subdomain is zero, the same flow is qualified to be harmonic.

a harmonic flow can be indirectly identified as the flow that cannot be explained by the available data inside the given domain. Such an interpretation is completely data-driven, and makes no assumptions about unknown external information.

This indirect interpretation of harmonic flow has an important advantage. Currently, there exists no technique that can use the divergence and curl of a given flow to determine whether it "contains" a nonzero harmonic flow or not. However, considering the presented interpretation, it becomes possible to implicitly identify any harmonic flow with respect to a given domain. Using this interpretation of harmonic flows, the natural HHD obtains uniqueness without assuming any boundary conditions a priori, and the results presented later in this chapter demonstrate that it is more suited for the purpose of data analysis, since it does not assume potentially inaccurate information.

3.2 The Natural Helmholtz-Hodge Decomposition

As mentioned above, the natural HHD decomposes a flow into internal and external influences. To understand the meaning of internal/external influences for a given domain, we refer to the theory of potential functions, discussed in Section 2.1. Every point (inside and outside the domain) with a nonzero energy source creates a potential at every other point. In the context of the HHD, the divergence and rotation fields of a given vector field act as collections of energy sources for divergent and rotational potentials, respectively. These potentials are used to compute the irrotational and incompressible components of the HHD, and thus, every point (inside and outside the domain) with a nonzero divergence and rotation influences these two components, respectively. Furthermore, as discussed in the previous section, harmonic functions are equivalent to external influences, meaning that a harmonic potential always represents external influences, and external influences always create a harmonic potential. These intuitions of internal and external influences are used to formally define the natural HHD.

Definition 3.1 (The natural HHD)**.** The *natural Helmholtz-Hodge decomposition*,

$$\text{HHD}^* : \vec{V} = \vec{d}^* + \vec{r}^* + \vec{h}^*,$$

is obtained by separating the flows due to internal and external influences, such that the *natural divergent* ($\vec{d}^*$) and *natural rotational* ($\vec{r}^*$) components represent the flows influenced by the divergence and rotation of $\vec{V}$ *inside* the domain. Consequently, the *natural harmonic* component ($\vec{h}^*$) is the flow influenced only by the exterior (boundary) of the domain.

3.2.1 Construction and Uniqueness

Given a smooth n-dimensional vector field $\vec{V} : \Omega \to \mathbb{R}^n$ for $n = 2, 3$, where Ω is a bounded subset of $\mathbb{R}^n$, the goal is to interpret and compute the natural components of $\vec{V}$. Using $\vec{V}$, define another smooth vector field on infinite space, $\vec{V}' : \mathbb{R}^n \to \mathbb{R}^n$, such that

- $\vec{V}'(\mathbf{x}) = \vec{V}(\mathbf{x})$ for all $\mathbf{x} \in \Omega$, and
- $|\vec{V}'(\mathbf{x})| \to 0$ as $|\mathbf{x}| \to \infty$.

By definition, $\vec{V}'$ captures the divergence and rotation of $\vec{V}$ in Ω, that is, $\nabla \cdot \vec{V}' = \nabla \cdot \vec{V}$ and $\nabla \times \vec{V}' = \nabla \times \vec{V}$ in Ω, and also provides a smooth flow on $\mathbb{R}^n$ that vanishes at infinity.

In order to compute the natural HHD of $\vec{V}$, the first step is to study the HHD of $\vec{V}'$. This indirect approach of using $\vec{V}'$ has two main advantages: (1) the HHD of $\vec{V}'$ is unique by definition, since it is defined on an infinite domain, and, hence, does not contain a harmonic component, and (2) it allows the use of the free-space Green's function, G_∞, which is trivially defined in closed form (see Expressions (2.4)). The HHD of $\vec{V}'$ is

$$\vec{V}' = \nabla D' + \nabla \times \vec{R}',$$

where the potentials D' and $\vec{R}'$ are the solution to the following two Poisson equations:

$$\begin{aligned} \Delta' D &= \nabla \cdot \vec{V}' && \text{in } \mathbb{R}^n, \\ \vec{\Delta} \vec{R}' &= -\nabla \times \vec{V}' && \text{in } \mathbb{R}^n. \end{aligned}$$

Using G_∞, D' and $\vec{R}'$ can be computed in closed-form as

$$\begin{aligned} D'(\mathbf{x}_0) &= \int_{\mathbb{R}^n} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla \cdot \vec{V}'(\mathbf{x})\ \mathrm{d}\mathbf{x} && \mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^n, \\ \vec{R}'(\mathbf{x}_0) &= -\int_{\mathbb{R}^n} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla \times \vec{V}'(\mathbf{x})\ \mathrm{d}\mathbf{x} && \mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^n. \end{aligned} \tag{3.1}$$

The integral solution presented above computes the potentials at a point $\mathbf{x}_0 \in \mathbb{R}^n$ due to the sources defined at all $\mathbf{x} \in \mathbb{R}^n$. Given a bounded domain $\Omega \subset \mathbb{R}^n$, the potential computation can be split into two parts—inside Ω and outside Ω. Furthermore, since $\nabla \cdot \vec{V}'(\mathbf{x}) = \nabla \cdot \vec{V}(\mathbf{x})$ and $\nabla \times \vec{V}'(\mathbf{x}) = \nabla \times \vec{V}(\mathbf{x})$ for $\mathbf{x} \in \Omega$, the Expressions (3.1) can be rewritten as

$$\begin{aligned} D'(\mathbf{x}_0) &= \int_{\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla \cdot \vec{V}(\mathbf{x})\ \mathrm{d}\mathbf{x} + \int_{\mathbb{R}^n \setminus \Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla \cdot \vec{V}'(\mathbf{x})\ \mathrm{d}\mathbf{x} && \mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^n, \\ \vec{R}'(\mathbf{x}_0) &= -\int_{\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla \times \vec{V}(\mathbf{x})\ \mathrm{d}\mathbf{x} - \int_{\mathbb{R}^n \setminus \Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla \times \vec{V}'(\mathbf{x})\ \mathrm{d}\mathbf{x} && \mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^n, \end{aligned} \tag{3.2}$$

where the integrals $\int_\Omega$ and $\int_{\mathbb{R}^n \setminus \Omega}$ represent the influence of the interior and the exterior with respect to Ω, respectively; these integrals are defined for the given field $\vec{V}$ and defined field $\vec{V}'$, respectively.

Furthermore, D' and $\vec{R}'$ are defined over $\mathbb{R}^n$, whereas the final goal is to compute the natural HHD of $\vec{V}$ in Ω. Therefore, the Expressions (3.2) can be restricted to $\mathbf{x}_0 \in \Omega$. Finally, since, by definition, the natural potentials represent only the internal influences, the following expressions provide formal definitions to the natural potentials D^* and $\vec{R}^*$:

$$\begin{aligned} D^*(\mathbf{x}_0) &= \int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla\cdot\vec{V}(\mathbf{x})\ \mathrm{d}\mathbf{x} \qquad \mathbf{x}, \mathbf{x}_0 \in \Omega, \\ \vec{R}^*(\mathbf{x}_0) &= -\int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla\times\vec{V}(\mathbf{x})\ \mathrm{d}\mathbf{x} \quad \mathbf{x}, \mathbf{x}_0 \in \Omega. \end{aligned} \tag{3.3}$$

It is important to note that ignoring the second integral in each of the Expressions (3.2) does not change the divergence and rotation captured by the nonharmonic potentials. This is true since the integral $\int_\Omega(\ldots)$ defines a nonharmonic potential inside Ω, whereas the integral $\int_{\mathbb{R}^n\setminus\Omega}(\ldots)$ defines a harmonic potential in Ω. As a result, the natural components $\vec{d}^* = \nabla D^*$ and $\vec{r}^* = \nabla\times\vec{R}^*$ match the divergence and rotation of $\vec{V}$, and the natural harmonic component, computed as the remainder $\vec{h}^* = \vec{V} - \vec{d}^* - \vec{r}^*$, is both divergence- and rotation-free. Finally, it should be noted that the natural potentials D^* and $\vec{R}^*$ are uniquely determined by Expressions (3.3), and therefore, so are $\vec{d}^*$, $\vec{r}^*$, and $\vec{h}^*$. Hence, the natural HHD is unique.

Theorem 3.1. Given a vector field $\vec{V}$ on a bounded domain Ω, its natural Helmholtz-Hodge decomposition as defined by Definition 3.1 is unique.

The significance of the natural HHD is that the uniqueness can be obtained without assuming any boundary conditions on D^* and $\vec{R}^*$.

3.2.2 Interpretation of the Natural Harmonic Flow

By definition, the two vector fields $\vec{V}$ and $\vec{V}'$ are equal inside the domain Ω.

$$\begin{aligned} \vec{V}'(\mathbf{x}_0) &= \vec{V}(\mathbf{x}_0) && \mathbf{x}_0 \in \Omega, \\ \nabla D'(\mathbf{x}_0) + \nabla\times\vec{R}'(\mathbf{x}_0) &= \nabla D^*(\mathbf{x}_0) + \nabla\times\vec{R}^*(\mathbf{x}_0) + \vec{h}^* && \mathbf{x}_0 \in \Omega, \end{aligned}$$

Therefore, the natural harmonic flow $\vec{h}^*$ inside Ω can be expressed as

$$\begin{aligned} \vec{h}^* &= \nabla D'(\mathbf{x}_0) + \nabla\times\vec{R}'(\mathbf{x}_0) - \nabla D^*(\mathbf{x}_0) - \nabla\times\vec{R}^*(\mathbf{x}_0) \\ &= \nabla\left(D'(\mathbf{x}_0) - D^*(\mathbf{x}_0)\right) + \nabla\times\left(\vec{R}'(\mathbf{x}_0) - \vec{R}^*(\mathbf{x}_0)\right) \end{aligned} \tag{3.4}$$

Substituting Expressions (3.1) and (3.2) in Expression (3.4),

$$\vec{h}^*(\mathbf{x}_0) = \nabla\int_{\mathbb{R}^n\setminus\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla\cdot\vec{V}'(\mathbf{x})\ \mathrm{d}\mathbf{x} - \nabla\times\int_{\mathbb{R}^n\setminus\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla\times\vec{V}'(\mathbf{x})\ \mathrm{d}\mathbf{x} \quad \mathbf{x} \in \mathbb{R}^n.$$

This expression means that the natural harmonic component of $\vec{V}$ can be constructed using the divergence and curl of a hypothetical flow $\vec{V}'$ on $\mathbb{R}^n$, and many such flows leading

to the same $\vec{h}^*$ can exist. This degree of freedom validates the understanding about a harmonic flow that its unique definition requires knowledge about the exterior. However, using the proposed internal-external split, defining a particular $\vec{V}'$ is not required, but instead, the influence of the unknown can be determined indirectly in a data-driven manner. As expected, for computation of $\vec{d}^*$ and $\vec{r}^*$, no information about the exterior (or the boundary) is needed, and therefore, the indirect approach to consider an infinite vector field is only a thought experiment.

3.2.3 Quantification of the Harmonic Flow

Although harmonic flows are well understood, currently there exist no techniques to determine whether a given flow on a given domain "contains" a harmonic piece or not. Since harmonic is defined as the flow that is both divergence- and rotation-free, at best, it is possible to evaluate only whether a given flow is rotation-free, divergence-free, or both (harmonic).

However, an earlier ignored but important property of harmonic flows is that they are equivalent to the influences that are external to the given domain. Utilizing this property, the internal-external split proposed in this work computes the harmonic flow indirectly, which makes it possible to, for the first time, answer an important question: *"Given a vector field $\vec{V}$ defined on Ω, how much harmonic flow does it contain?"*.

For a given flow $\vec{V}$ on Ω, its nonharmonic component, denoted as $\vec{v}^*$, can be computed using its divergence and rotation as follows:

$$\vec{v}^* = \left(\nabla \int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla\cdot\vec{V}\ \mathrm{d}\mathbf{x}\right) + \left(\nabla\times \int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \nabla\times\vec{V}\ \mathrm{d}\mathbf{x}\right),$$

and quantified using a vector norm ν that maps the space of vector fields to positive real numbers. By definition, ν must satisfy three properties:

- $\nu(\vec{a}) \geq 0$;
- $\nu(\vec{a}) = 0$ if and only if $\vec{a} = \vec{0}$;
- $\nu(\vec{a} + \vec{b}) \leq \nu(\vec{a}) + \nu(\vec{b})$.

Because the nonharmonic component $\vec{v}^*$ is guaranteed to not contain any external influences (harmonic), it is possible to quantify the amount of harmonic flow present in $\vec{V}$ indirectly. In particular, since $\vec{V} = \vec{v}^* + \vec{h}^*$, it follows that $\nu(\vec{V}) \leq \nu(\vec{v}^*) + \nu(\vec{h}^*)$. Hence, the lower bound on the norm of harmonic flow present in $\vec{V}$ with respect to Ω is

$$\nu(\vec{h}^*) \geq \nu(\vec{V}) - \nu(\vec{v}^*)$$

where both the terms in the RHS of this inequality can be uniquely determined.

In summary, the harmonic flow (external influences) present in a given flow can be quantified using a user-defined norm, such as the L_2 norm. The metric will be zero if and only if there are no external influences, and $\vec{V}$ can be uniquely decomposed into two natural nonharmonic components. On the other hand, a nonzero metric indicates that $\vec{V}$ is not directly useful to study internal features, and the natural HHD must be used to first dissociate the harmonic component. This quantification can also serve as a benchmark for different HHDs to assess the quality of the resulting components and their usability in applications requiring an open boundary.

3.2.4 Advantages and Limitations

Computing the natural components by distinguishing between the internal and external influences has several advantages. First, computation of the natural HHD depends only upon the flow inside the domain, and not on the relative alignment of the flow with its boundary. Notice that this approach also works for nonsimply-connected subsets of $\mathbb{R}^n$ (refer to Figure 3.5). Second, using the conceptual domain extension to $\mathbb{R}^n$, it becomes possible to utilize the free-space Green's function G_∞, which is simple and is known in the closed-form. Therefore, the computation on bounded domains can bypass the step of computing the finite-space Green's function G_Ω, which is much more involved, since it may not be known in closed-form for the given domain. Third, one could think of using a similar idea of extending the domain for existing techniques in order to reduce boundary artifacts. However, being global in nature, such techniques would depend upon the flow in the extended domain. On the other hand, the proposed approach is oblivious to how the field is completed in the extended domain, and does not depend upon the size and shape of the extended domain.

Finally, unlike the other techniques that use boundary conditions to obtain uniqueness, the computation of the natural HHD does not assume a decomposition of the boundary flow. Therefore, the fidelity of the flow is not compromised, and the components obtain the most "natural" values on the boundary.

The only limitation of the natural HHD is that it is not guaranteed to be L_2-orthogonal. However, orthogonality between the incompressible and irrotational components is important only for certain applications in fluid simulation, as it helps decouple the errors in the two components. For applications in visualization and analysis, orthogonality is not required. Instead, this work seeks more desirable properties such as flow topology, which can be attributed to the interior and exterior of the domain. Therefore, the natural HHD is suited to most applications in such areas.

3.2.5 Local Computation and Local Approximation

Besides the conceptual advantages of the natural HHD described above, it has some important practical benefits. The natural HHD is computed in a pointwise manner, and does not assume any boundary conditions. Consequently, considering a smaller area of interest $\Omega_1 \subset \Omega$, it is possible to obtain a *local decomposition* in Ω_1 by restricting the computation to Ω_1. This restriction computes the components in a smaller Ω_1 influenced by the divergence and rotation in the larger Ω. In contrast, using the state-of-the-art techniques [75, 163, 166, 204], which are global solutions, it is not possible to compute a local decomposition using global information.

Furthermore, it is also possible to obtain a *local approximation* to the global natural HHD by ignoring the flow outside Ω_1. This restriction computes the component flows inside Ω_1 due to the influence of Ω_1 instead of Ω. The error of this approximation is then given by the ignored term: $\int_{\Omega \setminus \Omega_1} \mathrm{G}_\infty (\ldots)$. However, in practice, this error is often negligible for two reasons. First, for turbulent flows, the external influences, for example, positive/negative divergences and clockwise/counterclockwise rotations, often cancel each other. Second, the Green's function is a distance-based weighting function whose value goes to zero as the distance increases. Therefore, for suitably chosen finite $\Omega_1 \subset \Omega$, the computation can be ignored for points in $\Omega \setminus \Omega_1$.

Both the local computation as well as the local approximation play crucial roles in terms of reducing computational costs, especially when Ω_1 is much smaller than Ω, as illustrated in Section 3.3.

3.2.6 Implementation Details

The expressions for natural potentials in 2D are

$$D^*(\mathbf{x}_0) = \frac{1}{2\pi} \int_\Omega \log(|\mathbf{x} - \mathbf{x}_0|)\ \nabla{\cdot}\vec{V}(\mathbf{x})\ \mathrm{d}\mathbf{x},$$
$$R^*(\mathbf{x}_0) = -\frac{1}{2\pi} \int_\Omega \log(|\mathbf{x} - \mathbf{x}_0|)\ \nabla{\cdot}\mathcal{J}\ \vec{V}(\mathbf{x})\ \mathrm{d}\mathbf{x},$$

where $\mathcal{J}$ is the $\pi/2$ rotation operator, and the incompressible component can be represented using a scalar potential R^*. The corresponding expressions in 3D are

$$D^*(\mathbf{x}_0) = -\frac{1}{4\pi} \int_\Omega \frac{\nabla{\cdot}\vec{V}(\mathbf{x})}{|\mathbf{x} - \mathbf{x}_0|}\ \mathrm{d}\mathbf{x},$$
$$\vec{R}^*(\mathbf{x}_0) = \frac{1}{4\pi} \int_\Omega \frac{\nabla{\times}\vec{V}(\mathbf{x})}{|\mathbf{x} - \mathbf{x}_0|}\ \mathrm{d}\mathbf{x}.$$

The computation of the natural HHD using these expressions requires (1) computation of the divergence, curl, and gradient; and (2) integration over the given domain. For both

these steps, any appropriate discretization can be used. For clarity of the presentation, the discussion is limited to bilinear vector fields on regular grids, and PC vector fields on triangulated domains. However, all concepts trivially extend to 3D and other interpolants.

In the following description, for a triangular domain $\mathcal{M}$, the PC vector field is defined on the faces of $\mathcal{M}$, and all scalar quantities are defined on the vertices of $\mathcal{M}$. Furthermore, for a regular grid $\mathcal{G}$ of size $[X \times Y]$ and grid spacings dx and dy, all scalar and vector quantities are defined on the vertices.

3.2.6.1 Computation of divergence, curl, and gradient. To define the differential operators on $\mathcal{G}$, the partial derivatives in each dimension are computed using central finite differences for the interior nodes, and forward/backward finite differences for boundary nodes. Using these partial derivatives, discrete approximations to the divergence, curl, and gradient operators are obtained.

For PC vector fields on $\mathcal{M}$, the finite element framework proposed by Polthier and Preuß [166] is used. The divergence (and curl) at a vertex is computed as the sum of dot products of the vector field with the normal (and tangent) along the boundary of the one-ring neighborhood of the vertex. The gradient of a scalar field defined at vertices of $\mathcal{M}$ is a vector field defined at its faces. For each face, it is the sum of the gradient of its barycentric coordinates weighted by the function value at the corresponding vertices.

3.2.6.2 Integration over the domain. Let $\mathbf{C}$, $\mathbf{E}$, and $\mathbf{I}$ be the set of vertices at the four corners, the boundary (not including the corners), and the interior of $\mathcal{G}$, respectively. Then, the integral of a function f over $\mathcal{G}$ is approximated using the 1D trapezoidal method over the x dimension, and subsequently over the y dimension, that is, by using the following expression:

$$\int_{\mathcal{G}} f(x,y) \approx \frac{dx\ dy}{4} \left(\sum_{\mathbf{v} \in \mathbf{C}} f_{\mathbf{v}} + 2 \sum_{\mathbf{v} \in \mathbf{E}} f_{\mathbf{v}} + 4 \sum_{\mathbf{v} \in \mathbf{I}} f_{\mathbf{v}} \right).$$

where $f_{\mathbf{v}}$ is the value of f at the vertex $\mathbf{v}$.

On the other hand, for every vertex $\mathbf{v} \in \mathcal{M}$, let $\text{Area}(\mathbf{v})$ denote the area of the corresponding cell in the Voronoi dual mesh (obtained by connecting the centroids of the triangles in $\mathcal{M}$). Then, the integral of a function over the domain is computed as

$$\int_{\mathcal{M}} f(x,y) \approx \sum_{\mathbf{v} \in \mathcal{M}} f_{\mathbf{v}}\ \text{Area}(\mathbf{v}).$$

Using the modules defined above, the integral solution to the Poisson equation is computed as weighted sum over all the vertices, making it an $O(n^2)$ algorithm, where n is the number of vertices in $\mathcal{M}$ or $\mathcal{G}$. However, the technique is trivially parallelizable since once

the divergence and rotation are available, computation at every vertex can be performed independently and in parallel.

3.3 Evaluation and Results

This section provides results of the natural HHD applied to some synthetic and simulated datasets. The aim is to first evaluate the accuracy and stability of the technique by comparing it with known component fields, and then to demonstrate its versatility and robustness by applying it to different physical flows. For comparisons, the LS-FEM approach of Polthier and Preuß [166] is chosen as the representative of HHD_{NP}—the HHD with NP boundary conditions—that is,

$$\vec{V} = \vec{d}_{\text{NP}} + \vec{r}_{\text{NP}} + \vec{h}_{\text{NP}},$$

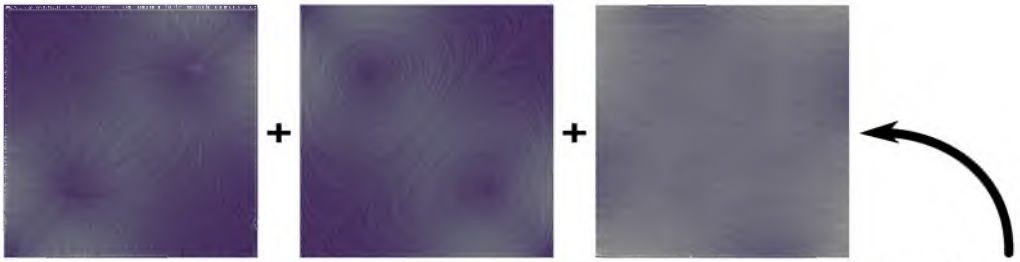
with boundary conditions: $\vec{n} \times \vec{d}_{\text{NP}} = 0$ and $\vec{n} \cdot \vec{r}_{\text{NP}} = 0$. Note that the majority of the differences between HHD_{NP} and HHD^* are due to the choice of boundary conditions, and not how the solution is computed. The visualizations in this section are created using line integral convolution (LIC) [20], color-mapped to represent vector magnitudes, where purple-white represents low-high values.

3.3.1 Comparisons with Analytical Fields

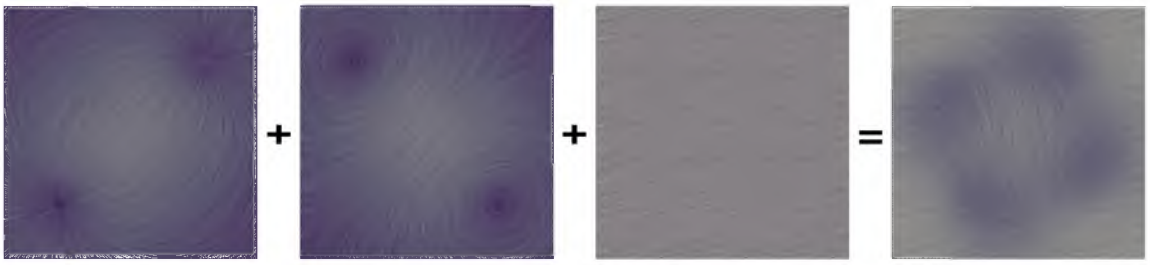
To evaluate the natural HHD, a synthetic vector field $\vec{V}$ is created by adding a divergent, $\vec{d}_{\text{true}}$, a rotational, $\vec{r}_{\text{true}}$, and a translational (harmonic), $\vec{h}_{\text{true}}$, flow, that is, $\vec{V} = \vec{d}_{\text{true}} + \vec{r}_{\text{true}} + \vec{h}_{\text{true}}$ as illustrated in Figure 3.3(b). Similar to the example given by Wiebel (see Figure 3.1), this field is contrived to represent a scenario requiring an open-boundary analysis, and, therefore, the standard boundary conditions are incompatible.

The "ideal" HHD of $\vec{V}$ should recover the three input ingredients accurately. However, as expected, the components of HHD_{NP} (refer to Figure 3.3(a)) show artifacts near the boundary. In particular, the flow magnitudes are near zero for both $\vec{d}_{\text{NP}}$ and $\vec{r}_{\text{NP}}$ near the corners of the domain, and the critical points in the rotational and divergent flows are shifted inwards. Furthermore, the corresponding harmonic flow $\vec{h}_{\text{NP}}$ does not reflect the added translation $\vec{h}_{\text{true}}$. In comparison, the components of the natural HHD (refer to Figure 3.3(c)) are virtually indistinguishable from the analytic components.

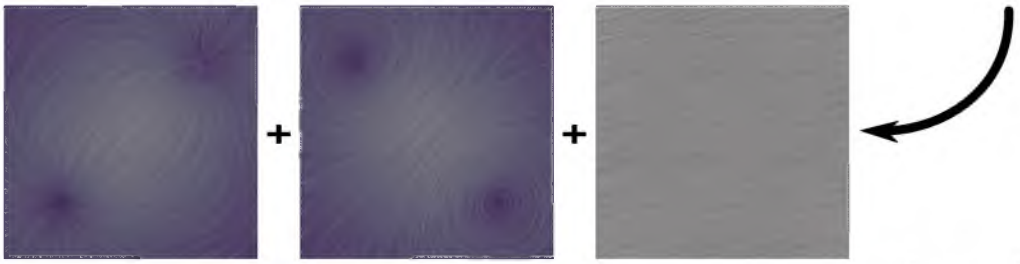
Since $\vec{d}$ and $\vec{r}$ are computed as gradient and curl of their corresponding potentials, their respective numerical rotation and divergence are always zero. Therefore, the divergence and rotation present in $\vec{h}$ are good indicators of the numerical quality of the decomposition. As shown by the first two sets of columns in Figure 3.4, the results of HHD^* closely match the



(a) The HHD using NP boundary conditions (HHD_{NP}): $\vec{V} = \vec{d}_{\text{NP}} + \vec{r}_{\text{NP}} + \vec{h}_{\text{NP}}$.



(b) Analytically created fields on domain $(-1, 1) \times (-1, 1)$. $\vec{d}_{\text{true}}$ has a source at $(0.5, 0.5)$ and a sink at $(-0.5, -0.5)$; $\vec{r}_{\text{true}}$ has two counter-rotating orbits at $(-0.5, 0.5)$ and $(0.5, -0.5)$; and $\vec{h}_{\text{true}} = \{0.5, 0.1\}$. All critical points were created such that their strengths decay exponentially with distance. The test field is obtained by adding these fields: $\vec{V} = \vec{d}_{\text{true}} + \vec{r}_{\text{true}} + \vec{h}_{\text{true}}$.



(c) The proposed natural HHD (HHD^*) without boundary conditions: $\vec{V} = \vec{d}^* + \vec{r}^* + \vec{h}^*$.

Figure 3.3. Comparison of (c) the natural HHD and (a) the HHD with NP boundary conditions with (b) analytic flows. The results of natural HHD are much closer to the ground truth.

expected results, whereas the results of HHD_{NP} show large deviations. To further compare the two decompositions, the amount of external influences in the corresponding components has been quantified. The third set of columns shows the L_2 norm for $|\vec{h}^*|$, $|\vec{h}_{\text{true}}|$, and $|\vec{h}_{\text{NP}}|$, and indicates that the natural HHD extracts almost all of the harmonic flow into $\vec{h}^*$. The last two sets of columns show the amounts of nonharmonic and harmonic flows present in the obtained divergence-free fields, denoted as $\vec{rv}$ and $\vec{rh}$. These fields are obtained through the natural HHD of the rotational fields obtained by the different techniques. The three columns for $\vec{rv}$ almost match, meaning that the net amount of rotational flow (with respect to L_2 norm) is conserved in all the rotational components. However, there exists a large amount of harmonic flow in $\vec{r}_{\text{NP}}$. In contrast, the harmonic flow contained in $\vec{r}^*$

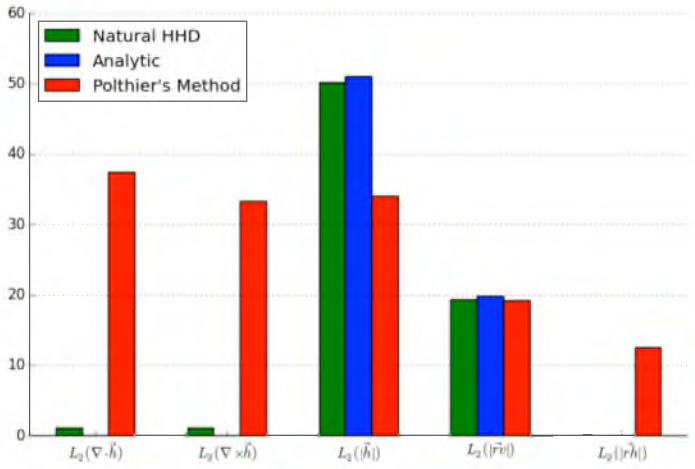


Figure 3.4. Quantitative comparison of the natural HHD and the HHD with NP boundary conditions with analytic flows. The figure shows that the natural components obtained through the natural HHD capture the analytic fields better.

is unnoticeable. The results on the obtained rotation-free fields (not shown in the figure) following a similar trend. These quantitative comparisons corroborate the visual impression that HHD* recovers the known solution significantly better than HHD_{NP}, whose results are not useful for applications requiring an open boundary.

Figure 3.5 shows the results of an experiment where an analytical flow ($\vec{V} = \vec{d}_{\text{true}} + \vec{h}_{\text{true}}$) was sampled on a regular grid, a rotated regular grid, and an unstructured annulus mesh.

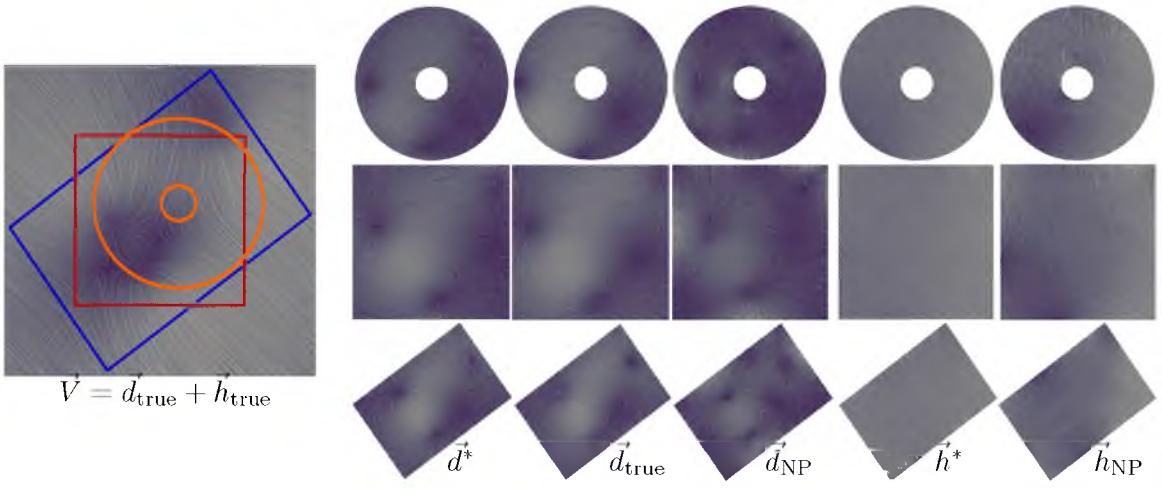


Figure 3.5. Comparison of the natural HHD and the HHD with NP boundary conditions with analytic flows sampled on different domains. The natural HHD ($\vec{d}^*$ and $\vec{h}^*$) contains significantly fewer artifacts than the HHD with boundary conditions ($\vec{d}_{\text{NP}}$ and $\vec{h}_{\text{NP}}$) as compared to the true components ($\vec{d}_{\text{true}}$ and $\vec{h}_{\text{true}}$). $\vec{h}_{\text{true}}$ is a translational flow $(1, -1)$, not shown in the figure.

Comparing the flows, one can observe that our results match the analytical flow much more closely. In particular, the fields $\vec{d}^*$ and $\vec{h}^*$ reflect the best estimates of $\vec{d}_{\text{true}}$ and $\vec{h}_{\text{true}}$, given the data inside the domain. On the other hand, $\vec{d}_{\text{NP}}$ and $\vec{h}_{\text{NP}}$ computed using NP boundary conditions impose boundary flow, and, therefore, show major deviations, especially near boundaries.

3.3.1.1 Stability. The next two experiments were designed to compare the stability of both techniques to sampling conditions and directions of the harmonic flow. First, a rotating field was sampled on a fixed structured domain. It was noticed that the proposed approach is virtually invariant to the sampling whereas HHD_{NP} continuously forces alignment with the boundary, causing significant artifacts. In the second experiment, the direction of the added harmonic flow was varied continuously. As expected from Figure 3.3, the natural HHD remains unaffected, but HHD_{NP} is able to recover $\vec{h}_{\text{true}}$ only when it aligns with the boundary and produces significant global artifacts otherwise. An accompanying video that shows animations for both these experiments can be found at the following link: `http://www.sci.utah.edu/~hbhatia/dissertation/TVCG-naturalHHD-video.mp4`.

3.3.2 Results on Simulated Flows

Next, the natural HHD is used to decompose a number of different simulated flows reflecting a variety of physical phenomena. In particular, the results presented in this section show how physical flows do not conform with traditional boundary conditions, and, thus, existing techniques are not applicable.

3.3.2.1 Flow behind the cylinder and cuboid. The data shown in Figure 3.6 is a single time-slice of the time-varying simulated flow behind a cylinder. In this setup, the flow is injected from the left boundary, and its behavior is observed behind the cylinder, which is located at the left boundary of the domain (see Data B.1.3). As seen in the figure, the natural HHD extracts the harmonic components completely, and $\bar{r}^*$ reveals the vortices. $\vec{d}^*$ is zero since the original flow is incompressible. Note how $\bar{r}^*$ is not aligned with boundary—a result not feasible with traditional techniques. Especially, note that the vortices in this data move left to right as time progresses. Therefore, depending upon the current value of time, the alignment of the flow inside the rightmost vortex changes with the right boundary of the domain, thus creating serious time-dependent artifacts. The computation of the natural HHD for this $[400 \times 50]$ data took about 1.06 seconds.

Figure 3.7 shows the results of the natural HHD on a similar 3D flow. The data represents the flow behind a cuboid, where the flow is injected from one direction, and its behavior is observed behind the obstacle. Notice that $\bar{r}^*$ reveals the vortical structures, and is not

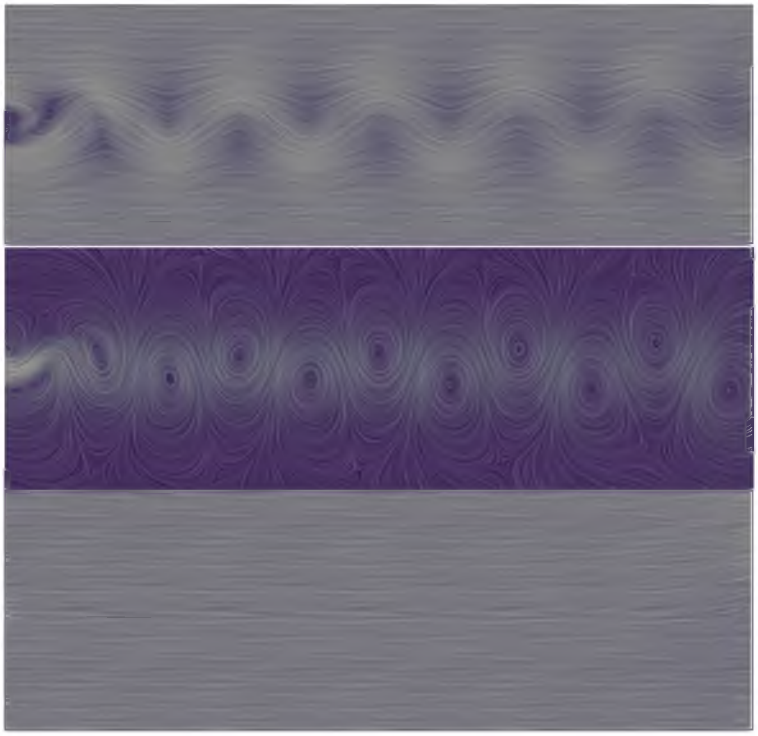


Figure 3.6. The natural HHD of the flow behind a cylinder (top). $\bar{r}^*$ reveals the von Kármán vortex street, and is not tangential with the boundary (middle). $\vec{h}^*$ represents the background flow (bottom).

compatible with the standard boundary condition, whereas $\vec{h}^*$ captures the background flow. The decomposition of this [101 × 101 × 101] flow took about 120 seconds with 144 processes in parallel.

3.3.2.2 Lifted ethylene jet flame. The second example is a direct numerical simulation of a turbulent lifted ethylene jet flame [222] (see Data B.3.1). Unlike the previous dataset, this is a compressible and highly turbulent flow. The fuel is injected on the bottom of the domain, creating a strong harmonic flow towards the top. Figure 3.8 shows one snapshot of a 2D slice from the center of the 3D flow. Both $\bar{r}^*$ (Figure 3.8(b)) and $\vec{d}^*$ (Figure 3.8(c)) are highly complex, not aligned with the boundary, and show some surprising structures. In particular, $\bar{r}^*$ shows two global counter-rotating vortices rather than a streak of smaller vortices one may have expected. Finally, $\vec{h}^*$ (Figure 3.8(d)) reveals an elliptical shape reflecting the nonconstant velocity profile imposed by the simulation. The decomposition of this [800 × 2025] data took about 290 seconds with 144 processes in parallel.

3.3.2.3 Jet in cross-flow. The next dataset represents a simulation of a jet in cross-flow, which is a fundamental flow phenomenon in many engineering applications [60, 73], for

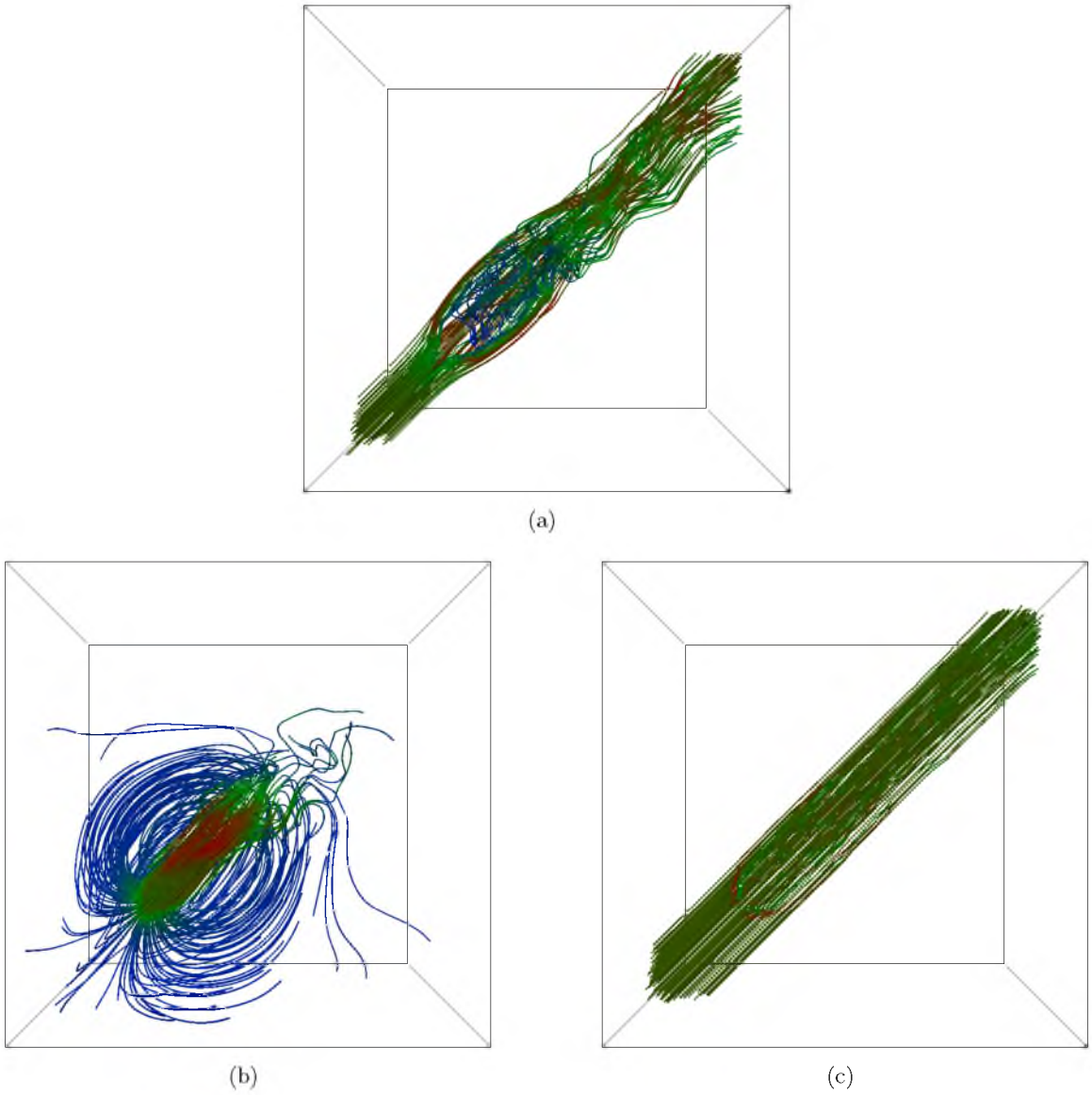


Figure 3.7. The natural HHD of the flow behind a cuboid (a). The figure shows streamlines in the interesting regions. (b) $\bar{r}^*$ reveals the vortical structures, and (c) $\vec{h}^*$ shows the translation present in the flow (right).

example, film cooling of turbines, fuel injections, and dilution jets in gas-turbine combustors. The experimental setup contains injection of flow through a jet at the bottom in the presence of a strong background transverse flow, the cross-flow (see Data B.3.2).

To simplify the illustration, a 2D slice is taken through the center of the 3D flow, such that the cross-flow is directed from left to right, and the jet appears at the bottom. Figure 3.9 compares the rotational fields obtained through HHD_{NP} and the natural HHD. The topological skeletons of these flows are computed by decomposing the domain into

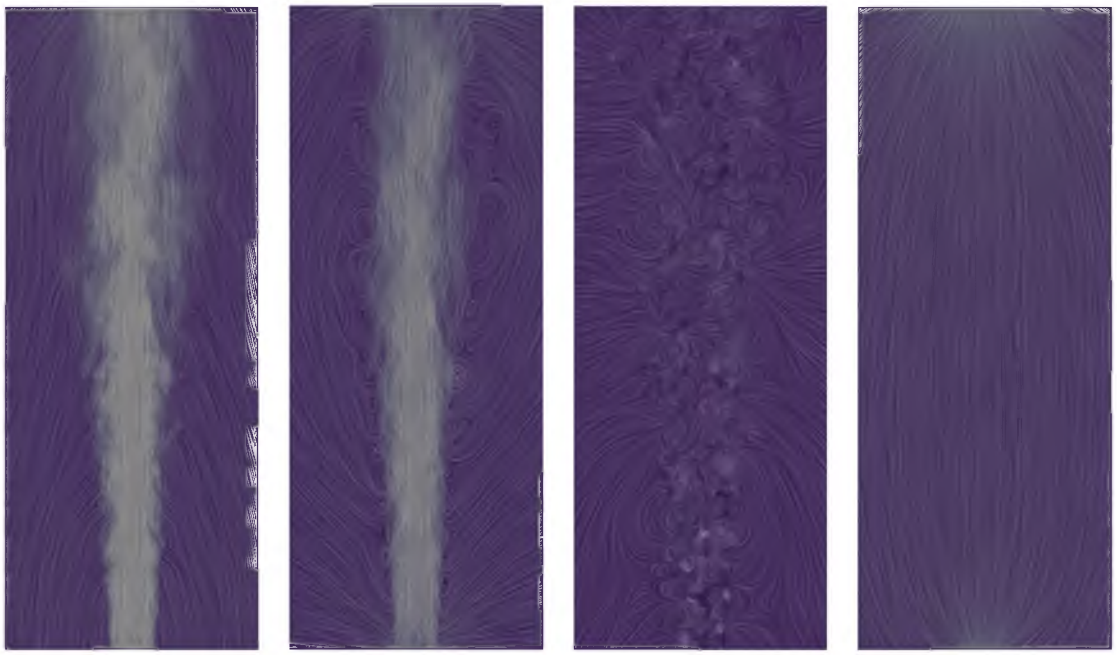


Figure 3.8. The natural HHD of the flow at the center of a lifted ethylene jet flame. (a) Flow at the center of a lifted ethylene jet flame [222]; (b) $\vec{r}^*$ showing two global counter-rotating vortices encapsulating smaller structures; (c) $\vec{d}^*$ showing a large number of sources and sinks related to volume changes; and (d) $\vec{h}^*$ computed as the residual showing an elliptical flow structure around the flame center.

nonoverlapping regions using the critical points and the saddle separatrices of the flow. Since the topological skeletons of the two rotational fields are significantly different, the important question is which of these flows corresponds to the expected combustion phenomena. Through scientific insights of domain experts, it is known that the small vortices near the top-left and bottom-left corners of $\vec{r}_{\text{NP}}$ are unphysical, since the underlying physical model expects vortices to be generated behind the jet only. Furthermore, note how $\vec{r}^*$ successfully captures the behavior of jet (green color), which is expected to rise up and go left to right (compare with the illustration shown in Appendix B). On the other hand, $\vec{r}_{\text{NP}}$ shows the jet (brown color) to follow a circular path that does not leave the domain from the right side, but comes back to the jet's point of entry. This behavior is also deemed unphysical by domain scientists. Thus, imposing parallel flow on the boundary produces false features (vortices) in the analysis.

These results demonstrate that the flow obtained through the natural HHD captures the physical model successfully, and it becomes possible to make enquiries about the accuracy or uncertainty of the computation. However, the flow obtained by using the boundary

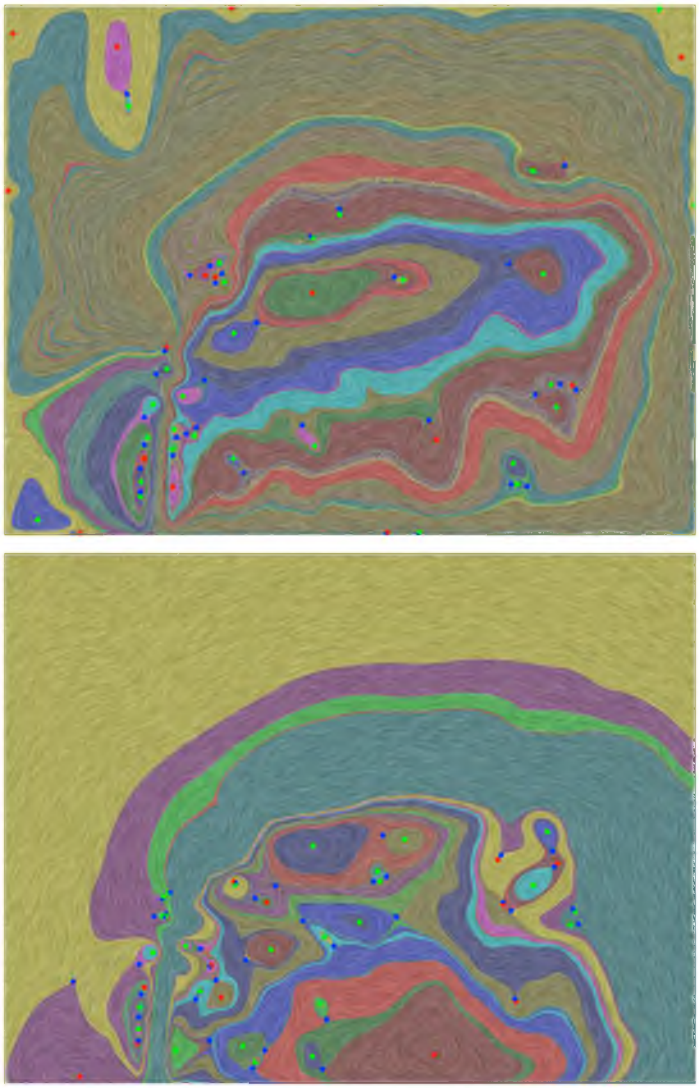


Figure 3.9. Comparison of the natural HHD and the HHD with NP boundary conditions on a 2D slice of the jet in cross-flow. The rotational fields $\vec{r}_{\text{NP}}$ and $\vec{r}^*$ are shown in the top and the bottom, respectively. Unphysical rotational structures are observed at the bottom-left and top-left corners in $\vec{r}_{\text{NP}}$ caused due to imposing parallel flow.

conditions does not respect the underlying physical model, and cannot be used to derive insights. Furthermore, the concerns of accuracy and/or uncertainty do not even make any sense in this case since the flow represents a different physical phenomena.

3.3.2.4 Oceanic currents. The final example is the surface flow taken from a 3D simulation of global ocean currents [133] (see Data B.3.3), as shown in Figures 3.10, with the focus on a $\Omega_1 = [200 \times 200]$ region in the South Pacific ocean. To demonstrate the practical utility of the local computation and the local approximation of the natural HHD, the decomposition for Ω_1 is computed using the data from concentric grids $\Omega = \mathcal{G}_N$. Figure 3.11

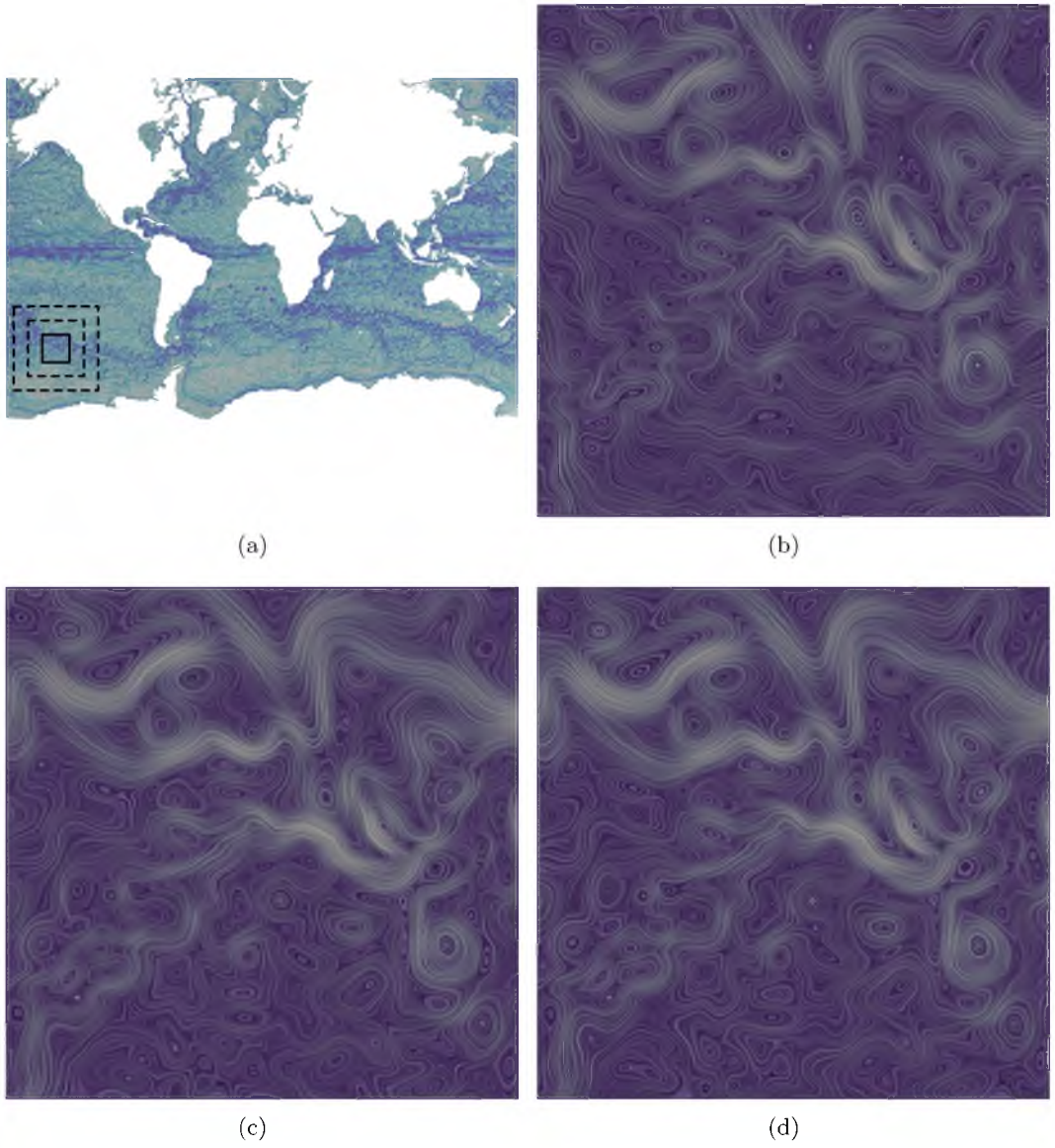


Figure 3.10. Local approximation of the natural incompressible component of the global oceanic flow, computed for the innermost tile, $\mathcal{G}_{200}$, shown in (a) by separately computing over three concentric tiles: (b) $\mathcal{G}_{200}$, (c) $\mathcal{G}_{400}$, and (d) $\mathcal{G}_{600}$.

shows a quantitative comparison of the results when N is increased from 200 to 600. It should be noted that whereas the time needed for the computation grows quadratically, the L_2 and L_∞ norms, and hence the decomposition, already seem to have converged. As a result, in order to analyze the flow in Ω_1, the natural HHD provides a local and faster technique, instead of using the existing global solutions, which would require the computation for the entire domain with over 5.3 million vertices. Figure 3.10 shows the rotational flows obtained for $\Omega = \mathcal{G}_{200}$, $\Omega = \mathcal{G}_{400}$, and $\Omega = \mathcal{G}_{600}$.

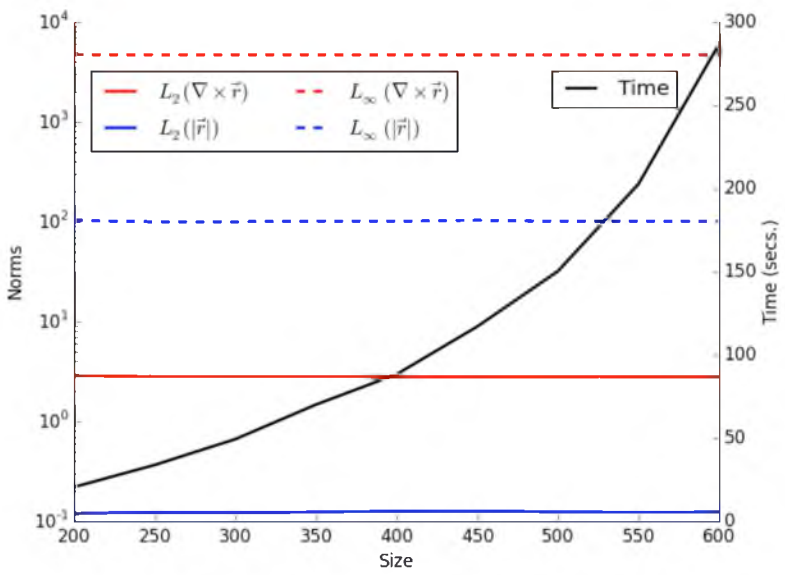


Figure 3.11. Quantitative evaluation of local approximation of the natural incompressible component of the global oceanic flow. Local computation for $\Omega_1 = \mathcal{G}_{200}$ using the data from $\Omega = \mathcal{G}_N$ with increasing N requires a quadratically increasing time (scaled to the right axis). On the other hand, the L_∞ and the normalized L_2 norms (log-scaled to the left axis) of $|\vec{r}|$ and $\nabla \times \vec{r}$ seem converged.

3.4 Summary

This chapter discusses the natural HHD, which is a new formulation to compute a unique decomposition of a vector field based on internal and external influences, allowing for an open-boundary analysis. The natural HHD uses the integral solution of the Poisson equation, decouples the decomposition from boundary conditions to avoid the typical boundary artifacts, and therefore produces significantly better results than state-of-the-art techniques. Furthermore, the decomposition can be computed locally, as well as in an approximate manner for faster, more targeted analysis of specific regions of interest. The proposed ideas allow, for the first time, quantification of the amount of harmonic flow in a given field. The computation of natural HHD opens new directions in support of various analysis techniques requiring independent study of internal and external influences.

Nevertheless, the full decomposition is computationally expensive and new approaches are necessary to make it applicable to large-scale 3D data. Furthermore, although the proposed approach works for a variety of spatial discretizations, it is limited to subsets of $\mathbb{R}^n$, and a similar approach for surfaces would be useful. Going forward, it is hoped that an easy to implement and reliable technique to compute the natural HHD will create new research interest in using it for various analysis tasks.

The natural HHD is an important step towards two of the main contributions of this work—computing local reference frames and reducing unsteady flows to sequences of steady flows. Next, Chapter 4 will discuss how the concepts presented in the current chapter can be used for achieving the final goal.

CHAPTER 4

COMPUTATION AND APPLICATIONS OF FLOW UNDER LOCAL REFERENCE FRAMES

The goal of this chapter is to discuss in detail the significance of local reference frames in the context of flows, especially unsteady flows. In particular, using the natural HHD presented in Chapter 3, this chapter discusses how to define a local reference frame, and how it can be used to reduce unsteady flows to sequences of steady flows.

Intuitively, in an unsteady flow, both the particles and the features are moving; typically, the speed of particles is too fast for the features, such that the latter fail to influence the former. In other words, before a feature can have a substantial influence on the nearby particles, those particles move on. As a result, when a single snapshot of an unsteady flow is seen, one often cannot identify features of interest. It is well known in the literature that such streamline-based analysis—which is instantaneous in nature—for example, using vector field topology [89], does not lead to meaningful features [129, 161, 162, 164]. Instead, the literature under consideration points out that such streamline-based analysis is applicable to unsteady flows only in a "meaningful" reference frame, where they are steady or near-steady in order for the corresponding features to be temporally coherent and physically meaningful. Intuitively, this different reference frame is equivalent to "slowing down" the particles enough that the influence of features can be detected using streamlines.

Nevertheless, it is not yet clear what this "meaningful" reference frame is; in fact, this dissertation argues that it is dependent upon the type of analysis to be performed or the type of features to be extracted. In this context, uniformly moving reference frames are of particular importance. Since such frames do not have any acceleration, they do not alter the physical laws of Newtonian physics. For flows, a uniformly moving frame does not alter certain properties of the flow, such as those derived from its deformation tensor, which, as a gradient, ignores constant motion. As a result, almost all scalar indicators designed to detect vortices, such as the Q-[97], Δ-[30], and λ_2-[100] criteria, are derived from the deformation

tensor, and therefore are *Galilean invariant*, that is, they are invariant to uniform motion. Other properties of the deformation tensor have also been used to classify regions with specific flow behavior [28, 30]. Kasten et al. [105] and Sahner et al. [177] also extract vortex cores using acceleration magnitude, and λ_2 vortex indicator, respectively—both of which are Galilean-invariant quantities. Assuming linearly moving critical points, Perry and Chong [161] also stressed the importance of Galilean invariance, stating that using a uniformly moving frame is sufficient in most cases for feature detection.

However, Galilean invariance does not suffice for all cases, since a nonuniformly moving object may be involved. Haller [82] proposed an "objective" definition of a vortex, which is invariant to more general motions of frames, namely, time-dependent translations and time-dependent rotations. Acknowledging the need for accelerating frames, Fuchs et al. [61] proposed to use an "unsteadiness" metric based on the material derivative of the Jacobian, whose minima were then used to extract coherent features.

This chapter presents new ideas that allow extraction of physically meaningful and temporally coherent features from unsteady flows, one time-step at a time, by computing new reference frames. It should be noted that both the indicator-based and the pathline-based approaches implicitly aim to process the data in a more useful reference frame, yet neither technique tries to construct such a frame. Instead, this work revisits the earlier arguments [129, 161, 162] and explicitly computes a more suitable reference frame. In particular, this chapter introduces the notion of *internal* reference frames, whose motion is represented by a harmonic flow; thus they are more general than uniformly moving frames. The internal frames can be computed in an efficient and massively parallel manner. Using several case studies, it is shown that the internal frames are indeed meaningful, and are able to extract local coherent features in large turbulent flows.

4.1 The Internal Reference Frame

An important consequence of the natural HHD (presented in Chapter 3) is that, by definition, the natural harmonic flow, $\vec{h}^*$, captures all external influences on the flow. This section discusses how this property makes $\vec{h}^*$ an excellent choice to describe a meaningful reference frame. Essentially, using the natural HHD, the motion of each particle in the flow can be expressed as a combination of two components, which are caused by (1) internal and (2) external influences. Thus, creating new particles, whose motion is described by $\vec{h}^*$, is equivalent to defining new observers that compensate for all external influences. This new "reference frame" allows studying the internal properties of the flow directly, and is

therefore called the *internal* reference frame.

Definition 4.1 (Internal Reference Frame)**.** A vector field is said to be analyzed in the *internal reference frame* if it is invariant to external influences (harmonic vector fields).

The term *"reference frame"* is used in a physical sense to characterize the state of motion of an observer, as done by Lugt [129], also referred to as an *observational frame* in the Newtonian relativity [112], where physical quantities are measured with respect to the state of motion of an observer. The term should not be confused with a very similar mathematical notion of a frame of reference that denotes a coordinate system. To understand the significance of internal frames, the reader can consider a passenger walking inside a moving train. A person standing outside the train represents a static reference frame, and cannot determine the speed of the walking passenger without knowing the train's speed. However, another passenger sitting inside the train can accurately describe the walking passenger's movements, since the sitting passenger represents a moving frame that already compensates for the train's motion. In the context of vector fields, the walking passenger represents the internal properties of the field; the outside observer represents the frame in which the field has been generated (through simulation or observation); and the second passenger is the internal frame. For a uniformly moving train, the reference frame is trivial, as each passenger experiences the same influence from the train. However, for more general motions, for example, a train on a curved track, each passenger will experience different influences depending upon its position within the train, and the radius and the center of the curve. As a result, each passenger (flow particle) must define a separate internal frame only for itself.

4.1.1 Transformation of the Flow into the Internal Frame

In order to transform a given flow into the internal frame, the definitions provided by the natural HHD are used. The natural irrotational ($\vec{d}^* = \nabla D^*$) and the natural incompressible ($\vec{r}^* = \nabla \times \vec{R}^*$) components of the flow $\vec{V}$ are defined using the natural potentials D^* and $\vec{R}^*$.

$$\begin{aligned} D^*(\mathbf{x}_0) &= \int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\nabla \cdot \vec{V}(\mathbf{x}) \right) \mathrm{d}\mathbf{x}, \\ \vec{R}^*(\mathbf{x}_0) &= -\int_\Omega \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\nabla \times \vec{V}(\mathbf{x}) \right) \mathrm{d}\mathbf{x}. \end{aligned} \tag{4.1}$$

The computation of these components requires only the divergence and rotation of $\vec{V}$ inside the domain. Since no boundary conditions are required to compute D^* and $\vec{R}^*$, these potentials contain no harmonic flow (no external influences). Using the expression of the

longitudinal and transverse components of the flow (see Expressions (2.16)), the natural potentials (Expressions (4.1)) can be expressed as

$$D(\mathbf{x}_0) = D^*(\mathbf{x}_0) - \oint_{\partial\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\vec{n} \cdot \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x},$$
$$\vec{R}(\mathbf{x}_0) = \vec{R}^*(\mathbf{x}_0) + \oint_{\partial\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\vec{n} \times \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x}.$$

Equating the natural HHD with the longitudinal and transverse decomposition, that is, $\nabla D^* + \nabla\times\vec{R}^* + \vec{h}^* = \nabla D + \nabla\times\vec{R}$, one gets

$$\begin{aligned} \vec{h}^* &= \nabla(D - D^*) \; + \; \nabla\times(\vec{R} - \vec{R}^*), \\ \implies \vec{h}^* &= \nabla {H_d}^* + \nabla\times\vec{H}_r^*, \end{aligned} \tag{4.2}$$

where

$$\begin{aligned} {H_d}^*(\mathbf{x}_0) &= -\oint_{\partial\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\vec{n} \cdot \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x}, \\ \vec{H_r}^*(\mathbf{x}_0) &= \oint_{\partial\Omega} \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \left(\vec{n} \times \vec{V}(\mathbf{x})\right) \mathrm{d}\mathbf{x}. \end{aligned} \tag{4.3}$$

Thus, the natural harmonic flow, which represents all external influences, can be computed using the two boundary integrals defined in Expressions (4.3).

To represent the internal frame, new particles must be created that move along the natural harmonic flow, $\vec{h}^*$. By the velocity addition formula of Newtonian relativity, such particles observe the original flow as

$$\vec{V}_{\text{int}}(\mathbf{x}, t) = \vec{V}(\mathbf{x}, t) - \vec{h}^*(\mathbf{x}, t), \tag{4.4}$$

where $\vec{V}_{\text{int}}$ is the flow in the internal frame. In other words, the flow $\vec{V}$ can be transformed into the internal frame by removing the natural harmonic component. For unsteady flows $\vec{V}(\mathbf{x}, t)$, this transformation can be applied to each time-step, leading to smooth $\vec{V}_{\text{int}}(\mathbf{x}, t)$ that can be used for extraction of meaningful features, as demonstrated later.

4.1.2 Internal Frame and Topology of the Domain

It is important to note that Equation (4.2) is valid only for simply-connected subsets of $\mathbb{R}^n$. For nonsimply-connected domains, there may exist other forms of harmonic flow induced by the topology of the domain, as explained by the Hodge-Morrey-Friedrichs decomposition [186]. As a result, it is not guaranteed that all harmonic flow can be represented by Equation (4.2). However, this is only a conceptual limitation and does not restrict the practical utility of the proposed technique for three reasons: (1) Most simulations impose no-slip boundary conditions at fluid-solid interfaces, that is, $\vec{V} = \vec{0}$. In

such cases, a smooth $\vec{V}_0$ can be defined on a simply connected $\Omega_0(\supset \Omega)$ by padding $\vec{V}$ with zeros in the holes of Ω. It can be shown that this padding does not influence the internal frame. Therefore, the internal frame can be computed easily for $\vec{V}_0$ on Ω_0 but used to analyze $\vec{V}$ on Ω. (2) Furthermore, many practical flows, such as the flow behind a cylinder (discussed in Section 4.2), do not contain any additional forms of harmonic, and therefore Equation (4.2) is applicable. (3) Finally, even when cases (1) and (2) are not applicable, the flow in the internal frame can be computed indirectly using the natural HHD, that is, $\vec{V}_{\text{int}} = \vec{d}^* + \vec{r}^*$.

4.1.3 Localized Internal Frame

Since the internal frame is defined on a pointwise basis depending only on the boundary data, it can be computed with respect to a boundary with an arbitrary shape for a given simply-connected subset of $\mathbb{R}^n$. This flexibility allows computing a *localized internal frame* for a more focused analysis in a smaller region of interest and to highlight local features. Furthermore, this technique also allows for a *local computation*, meaning that one can define the frame for a large domain, but compute it only in a smaller subset. The local computation further helps reducing the computational costs by focusing only on the regions of interest. Recall that $\vec{V}_{\text{int}}$ depends only upon the divergence and rotation of $\vec{V}$ inside the domain considered. Therefore, from the smoothness of $\vec{V}$, it follows that $\vec{V}_{\text{int}}$ will vary smoothly with respect to smooth changes in the domain considered.

4.1.4 Implementation Details

Computation of the flow in the internal frame is similar to that of the natural HHD, discussed in Section 3.2.6, and requires two modules: (1) computation of divergence, curl, and gradient; and (2) computation of boundary integrals of normal and tangential components of the flow. With the availability of these modules, the computation of the flow in the internal frame using Expressions (4.3) is straightforward and trivially parallelizable for a variety of spatial discretizations and/or interpolants. For brevity, only bilinear/trilinear flows defined on regular grids, and piecewise constant (PC) flows defined on triangulated meshes are discussed here.

For a regular grid $\mathcal{G}$ with vector samples defined on vertices, the normal and tangential components of the boundary flow can be defined trivially with respect to the exterior normal, once an orientation is assumed with respect to Cartesian coordinates. Since each "side" of $\mathcal{G}$ is also a regular grid, integration along the boundary can be approximated using the trapezoidal method applied successively in each dimension. Finally, finite differences can

be used to define the required differential operators.

For a triangulated domain $\mathcal{M}$ with a PC vector field defined on its faces, the normal and tangential components are computed with respect to the edges on the boundary, and boundary integrals are approximated by summing these components. The divergence (and curl) of the vector field is defined at the vertices of $\mathcal{M}$, and is computed as the sum of dot products of the vector field with the normal (and tangent) along the boundary of the one-ring neighborhood of the vertex [166]. The gradient of a scalar field defined at vertices of $\mathcal{M}$ is a vector field defined at its faces. For each face, it is the sum of the gradients of its barycentric coordinates weighted by the corresponding function values.

4.2 Evaluation and Results

This section presents the results of the experiments performed to evaluate the internal reference frame. First, the flow behind a cylinder is used to compare different frames. Since the resulting features of this flow are known, this data is used to demonstrate that the internal frame extracts features whose behavior is consistent with theory, and is indeed a distinguished frame. Next, the internal frame is used to explore complex flows resulting from combustion simulations.

To show that the internal frame allows representing unsteady flows as sequences of steady flows, this section shows that the features extracted from the individual time-steps of the unsteady flows using streamline-based techniques are meaningful. We use the vector field topology [89] as a representative of streamline-based techniques. Therefore, the critical points and the saddle separatrices of the flow are computed in a *consistent* manner (using the framework discussed in Part III). These features define the topological skeleton of the flow, which is used to decompose the domain into nonoverlapping regions. Most of the following figures show this topological decomposition by highlighting these regions in different colors overlaid on the image-based flow visualization (IBFV) [208]. In the remaining figures, the IBFV is overlaid with the colormap shown here to represent vector magnitudes, with blue-to-red representing low-to-high values. Blue spheres denote saddles in the flow, and red and green spheres denote clockwise and counterclockwise rotating vortices. The performance was noted for double-precision data and a serial algorithm running on a MacBook Pro with 2.8 GHz Intel Core 2 Duo processor and 8 GB RAM.

4.2.1 Simulated Flows

4.2.1.1 Flow behind a cylinder. The 2D flow behind a cylinder is expected to exhibit the well-known von Kármán vortex street, as illustrated in Figure B.3 (see Data B.1.3).

Using the flow simulated for the Reynolds number, $Re = 165$ and inlet horizontal velocity, $u = 1.1$, it is observed in Figure 4.1 that the topological decomposition of the flow in the simulation's reference frame is trivial, since the entire flow originates from the left boundary and exits from the right. However, the same analysis performed in the internal frame reveals the expected vortex street, as illustrated by cyclic arrows in the figure. This is a well-studied flow, typically used for illustrating vortex detection techniques, most of which rely on scalar indicators associated with thresholds [105]. The internal frame allows the computation of such regions by analyzing the flow directly from a single time-step, without losing information about it. For example, the internal frame also reveals the region of backward flow (highlighted by arrows) between the counter-rotating vortices, which induces drag on the cylinder, whereas the original frame obscures this behavior. Furthermore, neither the techniques based on scalar indicators nor the FTLE-based approaches are able to represent backward flow due to information loss during the corresponding transformations. Given the results obtained in the internal frame, it becomes possible to answer many important questions about the underlying flow phenomena: What is the spatial distribution of the vortices in the vortex street? What are the distances between any two vortices—both horizontally and vertically? What are the areas/volumes enclosed by these vortices? What are the strengths of these vortices? What is the shedding frequency of the vortices? How fast are these vortices moving? etc. Note that the existing techniques typically focus on

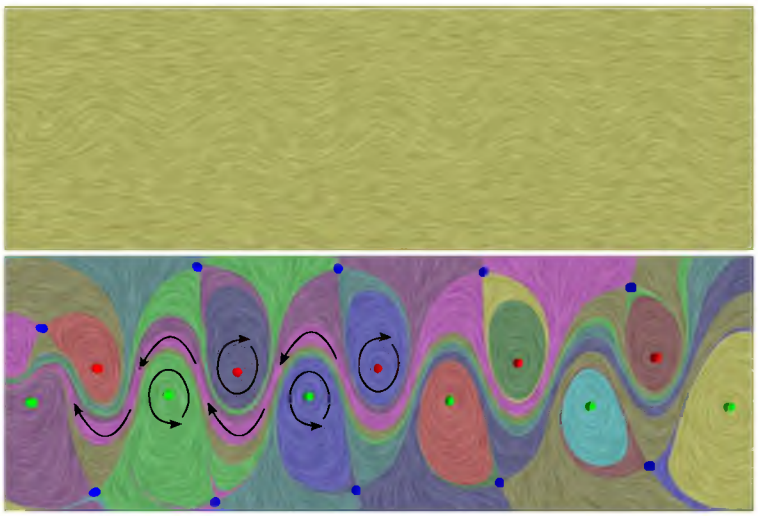


Figure 4.1. Comparison of the topological decomposition of the flow behind a cylinder in the simulation's (top) and the internal frame (bottom). The result in the simulation's frame is trivial, whereas the same analysis in the internal frame reveals vortices with backward flow between them. The flow in the internal frame for this $[1450 \times 400]$ data took ≈ 43 sec. to compute.

one of these question, whereas the internal frame allows performing a holistic analysis of the phenomena.

To demonstrate that the internal frame is indeed a distinguished frame for analysis, the natural harmonic flow, which represents the motion of the internal frame, was computed, and its magnitude was increased and decreased. As illustrated in Figure 4.2, the analysis in faster or slower frames produces incorrect features, that is, the features that are inconsistent with theory. In particular, in the absence of any discretization asymmetries, the vortex street is expected to be vertically symmetric (with horizontal offsets) about the center of the cylinder at all times because both the cylinder and the inlet velocity are vertically symmetrical. However, for slower and faster frames, the instantaneous flows are observed to be moving, more or less, downward and upward, respectively. It should also be noticed that the amount of background flow exhibited in the slower frame is significantly reduced, whereas the faster frame exhibits inverse flow at the top of the domain.

Next, the same analysis is performed for the flows for increasing Reynolds numbers, $Re = 100$, 125, 140, and 165, corresponding to a fixed kinematic viscosity and $u = 0.92$, 0.98, 1.01, and 1.1. To verify the temporal coherence of the results, vortices are computed for the simulation time-range $[10, 22.5]$ consisting of 26 snapshots, one at a time. Figure 4.3 shows the trace of some of the vortices extracted from these snapshots using one spatial domain to encode time. As expected, the locations of the critical points track the inherent

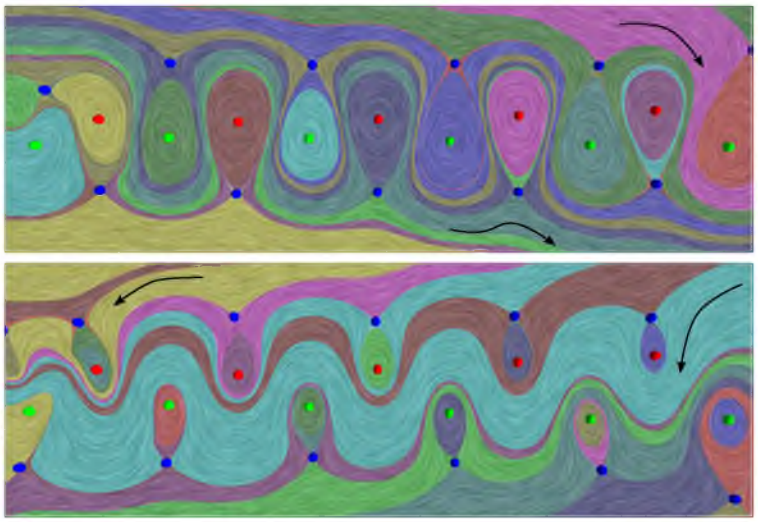


Figure 4.2. Topological analysis of the flow behind a cylinder in faster (bottom) and slower frames (top). Slowing down or speeding up the internal frame yields inconsistent results. The results for 10% changes to the internal frame clearly demonstrate that the flow is no longer vertically symmetrical.

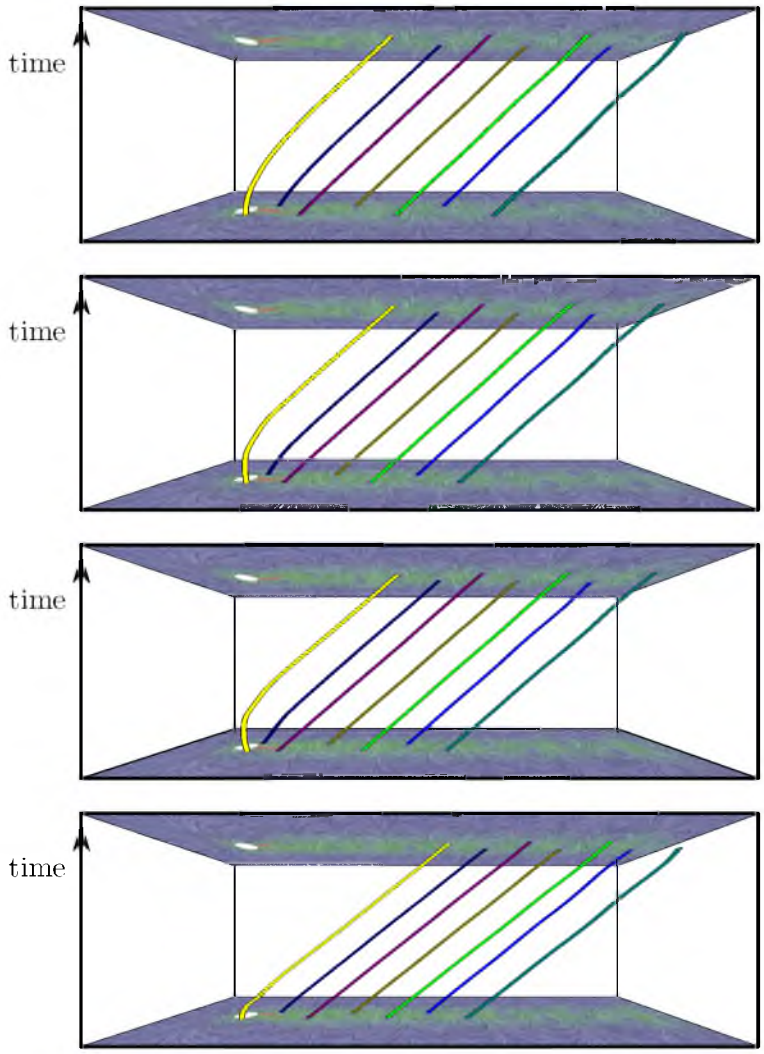


Figure 4.3. Vortex traces in the flow behind the cylinder. From top to bottom, with increasing Re, the speed of vortices increases as expected. To reduce clutter, only those vortex traces are shown that exist for the entire time-range.

motion of the vortices, allowing an accurate measurement of the speed of their motion. These results confirm that the vortices move slightly slower than ($\approx$ 91% of) the inlet velocity. In Figure 4.3, the speeds of vortices are ≈ 0.85, ≈ 0.89, ≈ 0.92, and ≈ 1. It should be noted that flows in the internal frame exhibit temporal stability, and so no actual tracking of the vortices was necessary to produce these results; instead the figures simply show the position of the vortex core at the given time-steps. It should also be noted that an approximation to shedding frequency computed in the internal frame also increases with increasing Reynolds number—a trend consistent with theory.

Finally, the topological analysis in the internal frame is evaluated against the particle-based and indicator-based techniques, both in the simulation's frame. Figure 4.4 shows

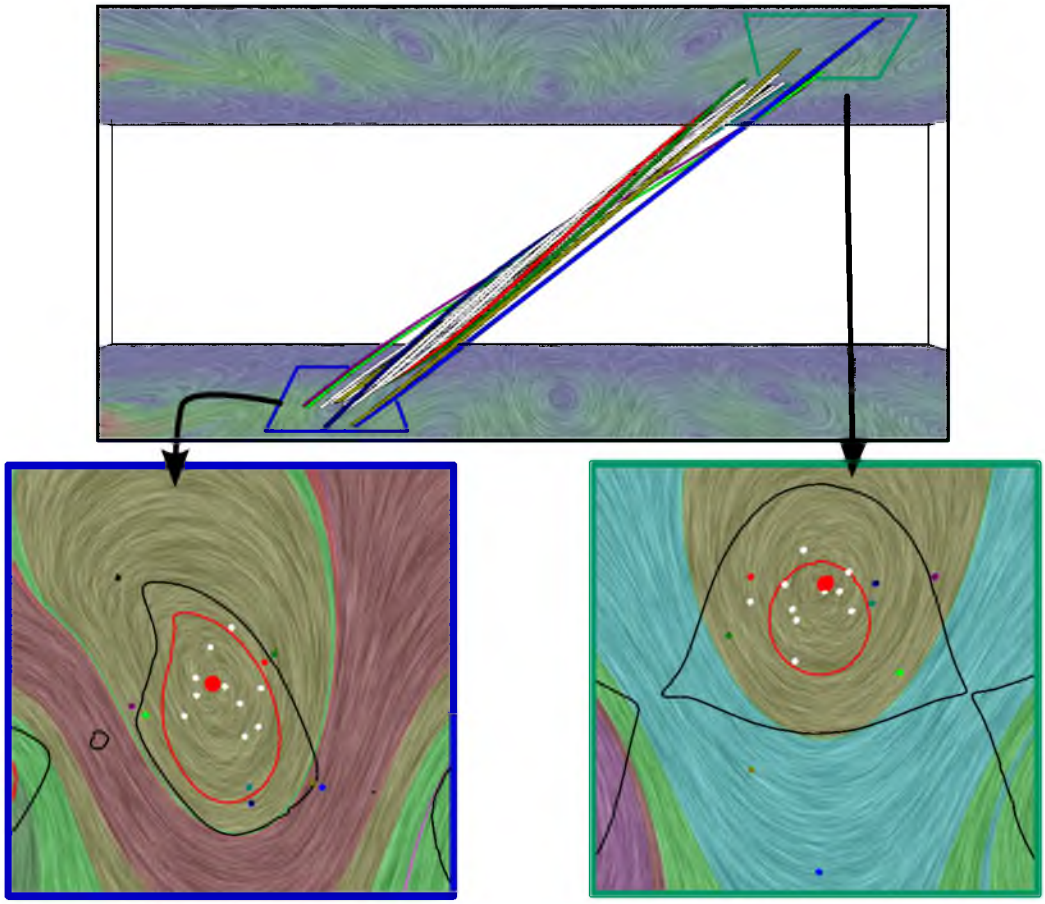


Figure 4.4. Comparison of the topological analysis in the internal frame with the indicator-based and pathline-based analysis in the simulation's frame. The analysis in the internal frame can capture spatiotemporal dynamics of pathlines (in the center) by analyzing the flow one time-step at a time. Left and right show topological analysis that successfully captures the start and end of these pathlines, useful for, for example, in identifying vortices. In contrast, the contours of Q-criterion, shown in black ($Q = 0$) and red ($Q = 0.1$), create false-positives and false-negatives.

a few pathlines computed in the simulation's frame, along with IBFV of the flow in the internal frame. On the left and the right, we zoom in to show the position of these particles at $t = 10$ and $t = 22.5$, overlaid on the topological decomposition computed in the internal frame, and the contours of Q-criterion [97] computed in the simulation's frame. The result asserts that the vortex identification using Q can create false-positives and false-negatives, for example, the olive and the green colored particles, respectively. From the figure, it can be noticed that the olive pathline does not "swirl" around the vortex core, whereas the green pathline does. In contrast, the topologically-identified vortex in the internal frame captures the true swirling behavior of the particles, from a single time-step and at much lower computational expense.

4.2.1.2 Lifted ethylene jet flame. Next, the direct numerical simulation of a turbulent lifted ethylene jet flame [222] is studied, which represents a compressible and highly turbulent flow (see Data B.3.1). In this case, fuel is injected from the bottom of the domain, creating a strong background flow towards the top. The natural harmonic flow, which represents the motion of the internal frame, reveals an elliptical shape (see Figure 4.5(b)) consistent with the nonuniform velocity profile (faster flow in the middle) imposed by the simulation. As a result, this flow is an example where nonuniformly moving frames cannot be compensated for using Galilean-invariant techniques. In contrast, the more general internal reference frame can successfully extract the hidden features in this flow.

As shown in Figure 4.5(c), analyzing a single snapshot of the flow in internal reference frame reveals two global counter-rotating vortices rather than a streak of smaller vortices

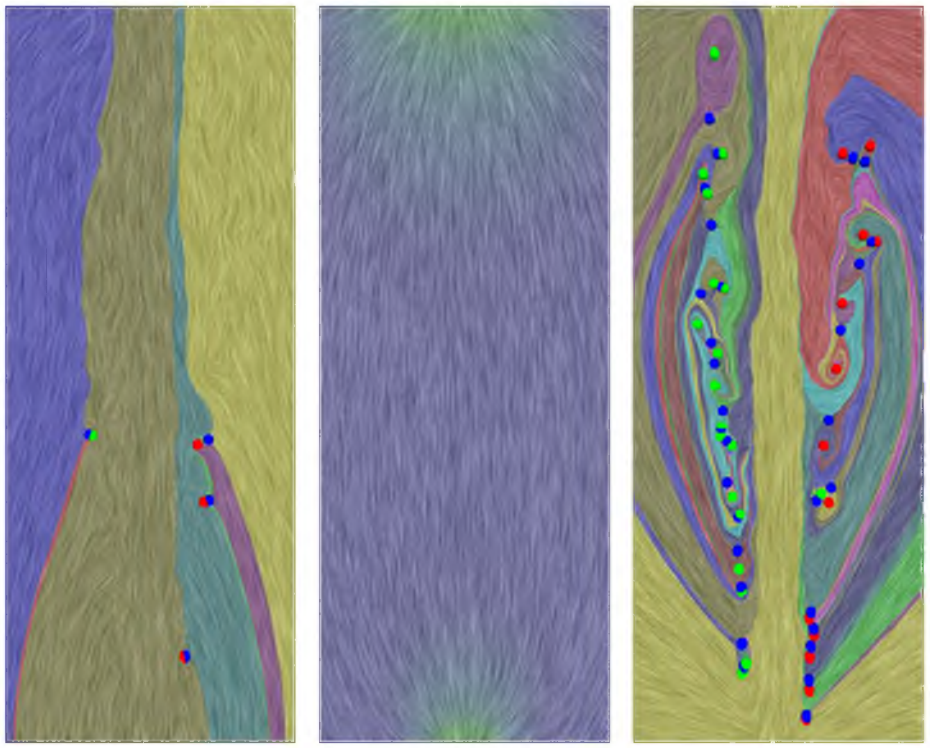


Figure 4.5. Comparison of the topological decomposition of the lifted ethylene jet flame in the simulation's (left) and the internal (right) reference frames. The natural harmonic flow (middle) describes the motion of the internal frame. Only 8 critical points are identified in the original frame, whereas 62 critical points describe the complex dynamics of this simulated flow in the internal frame. The internal frame for this $[800 \times 2025]$ data took ≈ 186 sec. to compute.

one may have expected. Instead, the smaller vortices are nested in these global rotating structures. The figure also highlights the flow that runs through the domain from bottom to top, along with all the smaller structures that exist on either side of this through-flow. As is clear from the figure, none of these intrinsic features are visible in the simulation's reference frame (see Figure 4.5(a)).

Figure 4.6 shows traces of vortices for time $[0.00174, 0.00175]$, confirming that the vortices are temporally coherent. Many vortices are small and cancel out with saddles in the simulation, but some that are stable and persist for the entire time-range are shown.

4.2.1.3 Jet in cross-flow. The next dataset is generated from a simulation of a jet in cross-flow [73] (see Data B.3.2). The goal of the experiment is to study different types of vortical structures created by the interaction of the burning jet with the cross-flow. For clarity in presentation, the results obtained in the internal frames are first shown for selected xz and yx slices. Since the cross flow is transversal to yz planes, the simulation's frame is already very similar to the internal frame. First, a xz plane is analyzed for $y = 550$, which cuts through the center of the jet. Figure 4.7 shows the topological decomposition of the flow in the original and the internal frames. Notice the vortical structures revealed in the internal frame. To emphasize the differences, the zoomed-in region surrounding the jet highlighting the presence of shear-layer vortices around the jet. Other vortices that cannot be observed in the simulation's frame are also revealed.

Next, the topological decomposition of the flow is studied in the internal frame for slices representing yx planes. As shown in Figure 4.8, interesting features are observed for slices $z = 50$ and $z = 350$, both of which cut through the jet transversally. The first slice is near the orifice, and shows the formation of two counter-rotating vortices in a nascent stage (highlighted as rotation). This visualization helps in observing how these vortices grow with

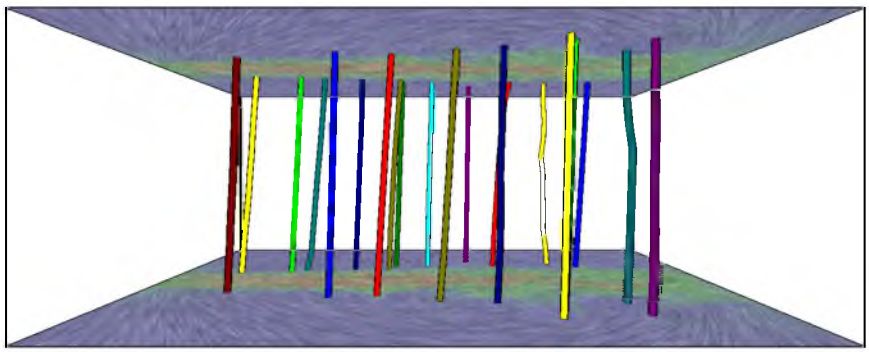


Figure 4.6. Vortex traces show the presence of stable rotational structures in the lifted ethylene jet flame.

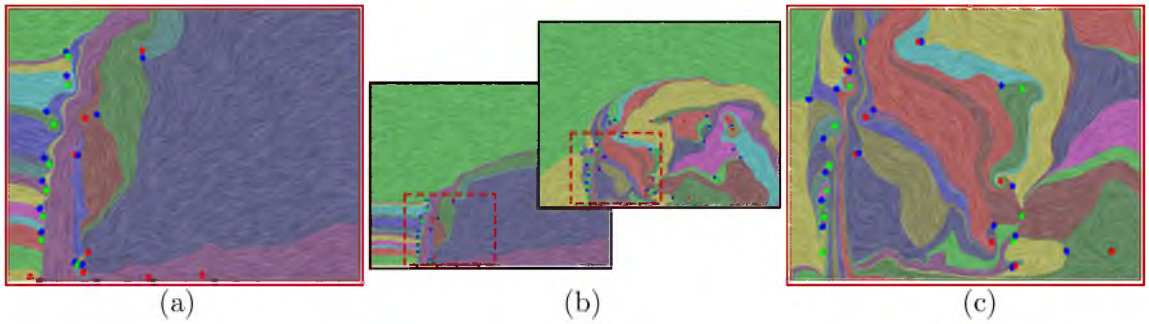


Figure 4.7. Topological decomposition of a xz slide of the jet in a cross-flow in the simulation's frame (b bottom, a) shows little structure as the interesting intrinsic phenomena are overshadowed by the background flow. The same topological analysis of the flow in the internal frame (b top, c) reveals the expected shear-layer vortices on the front of the flame as well as a wealth of other features.

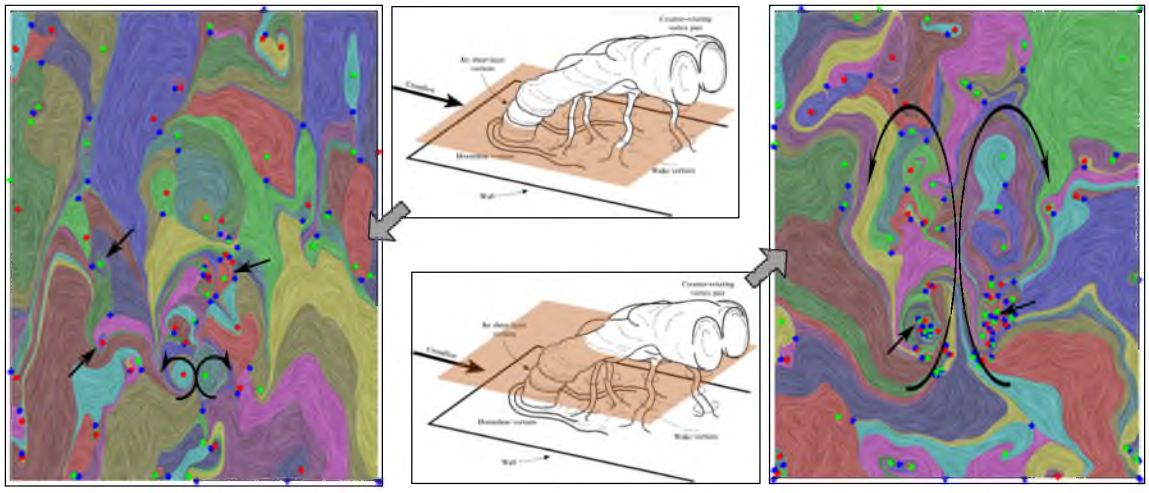


Figure 4.8. Topological decomposition of yx slices of the jet in cross-flow in the internal frame shows the interior of the jet near its orifice, and the evolution of jet with height. Observed wake vortices are marked with arrows.

increasing z. Other smaller vortices are observed above these structures, which appear to be the wake vortices (highlighted using arrows). As expected at a greater height ($z = 350$), one observes that these vortices stretch in the x-direction (vertical in the figure) as the plane cuts through a greater part of the vortices, while the wake vortices now appear below them.

4.2.1.4 Localized 3D computation. Finally, we present results on the 3D flow in the internal frame. We analyze a $[400 \times 200 \times 250]$ subset of the data, which was chosen by scientists as it covers the most interesting region of the flow around the jet's orifice. Figure 4.9(a) shows the natural harmonic flow computed for this region. Figures 4.9(b) and 4.9(c) show two views of the flow in the internal frame, highlighting different structures.

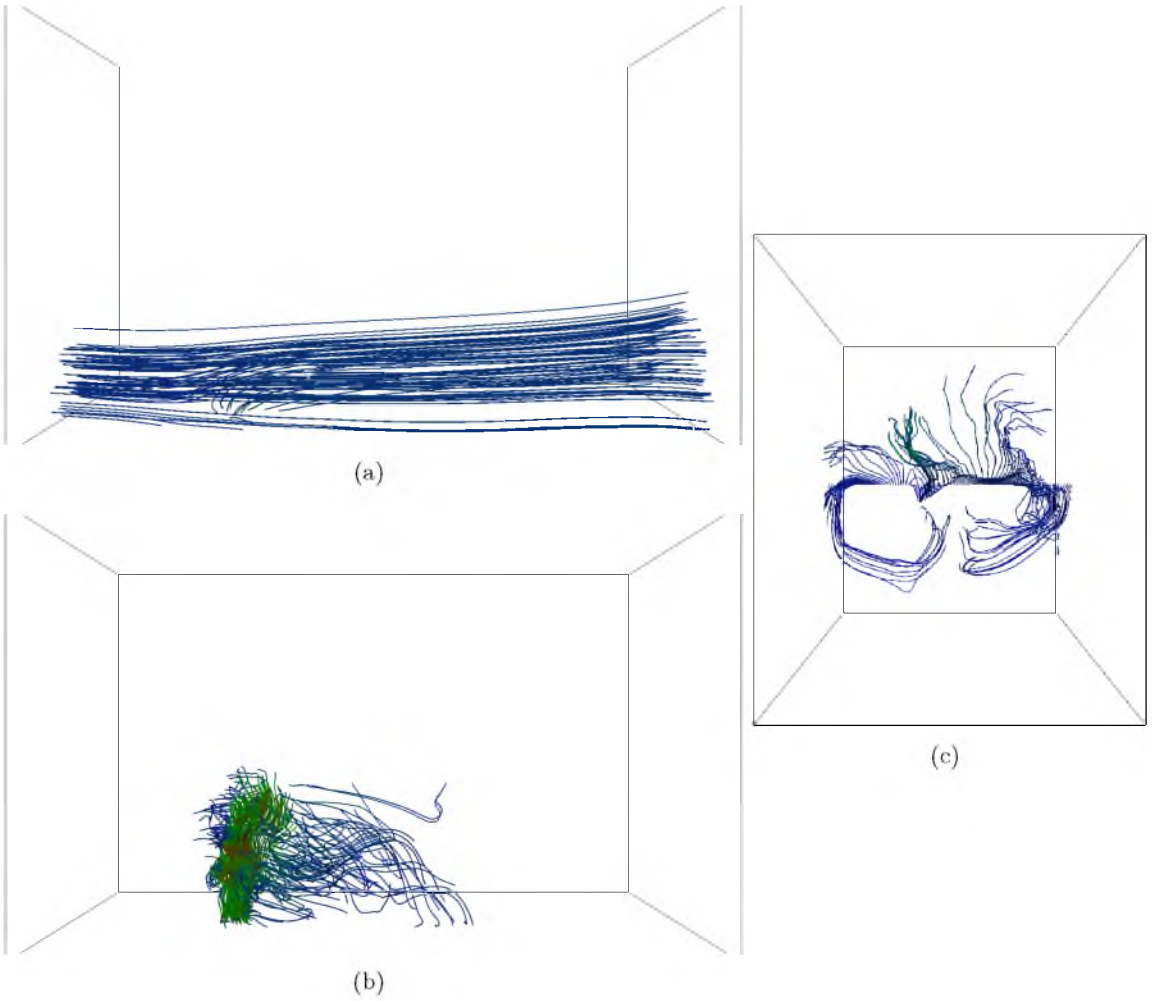


Figure 4.9. Localized analysis on a 3D subset of the jet in cross-flow. (a) The natural harmonic flow. (b) and (c) Flow in the internal frame highlighting features near the jet orifice (b) and the formation of counter-rotating vortex pairs (c).

The first view focuses on the jet and shows the vortices created around it, and the second shows the formation of the counter-rotating vortex pair, a structure notoriously difficult to extract. The challenge is that the vortical motion is significantly weaker than that of the smaller structures in the interior, which, therefore, dominate the former when traditional techniques are used. In fact, several of our scientific collaborators have remarked that this is the first time they have seen this structure visualized directly.

4.3 Summary

This chapter presents new ideas for computing local reference frames that allow extraction of localized and otherwise obscured features from vector fields. Using the natural HHD, a given flow is decomposed into appropriate components, where the resulting natural

harmonic component captures the external influences with respect to a given domain, and the remaining nonharmonic component represents the internal behavior. Equivalently, the nonharmonic component represents the flow in the internal reference frame, whose motion is represented by the harmonic component.

An important consequence of the internal reference frame is that it "slows" down the flow, which allows extraction of important features from individual time-steps. As a result, these ideas enable streamline-based analysis for unsteady flows, allowing a deeper impact on real problems of interest. The results of streamline-based analysis presented in this chapter demonstrate that the analysis in the internal frame captures the *spatiotemporal* dynamics of material particles. The ideas presented in this chapter simplify the analysis of unsteady flows by decoupling the spatial and temporal dimensions, so that a simpler spatial analysis can be performed on individual time-steps followed by a simpler temporal analysis. A possible future impact of this work is that by using meaningful reference frames, it may be possible to revive the abandoned idea of analyzing unsteady flows one time-step at a time.

Although there are examples where the existence of a distinguished frame is not yet known, for example, the double gyre [187] and the petri dish [218], the proposed frame addresses a large class of flows that represent physical phenomena in the presence of a background flow or around a moving object. In particular, the motion of the internal frames is represented by a (possibly time-varying) harmonic flow, thus capturing more general background flows than uniformly moving ones. It is hoped that the proposed work will foster new research towards computing more general frames, for example, rotational, that can help analyze other types of flows as well.

The proposed ideas make a large number of existing techniques for steady flow analysis directly applicable to unsteady flows. Consequently, the proposed work can open new horizons in in situ analysis of large-scale complex flows, where simulations are temporarily halted to analyze the current state and only the results are dumped. Especially with anticipated advances in compute power and lingering I/O bottlenecks, such analysis can enhance our ability to analyze data of ever-growing scale and complexity at much higher temporal resolutions.

CHAPTER 5

EVALUATION OF PATHLINES OF LARGE-SCALE SIMULATED FLOWS

One of the most common ways of analyzing unsteady flows is through pathline-based techniques. As the name suggests, such techniques compute the paths of massless particles over time as they are advected by the flow, and use these paths to extract and represent flow behavior. Common examples include the computation of pathline-based topology [203], and the detection of material boundaries using the finite-time Lyapunov exponent (FTLE) [80] field. Pathline-based approaches capture the behavior of particles over space and time, and, therefore, the resulting features are also spatiotemporally coherent.

Nevertheless, such techniques are computationally expensive, since they require computation of a very large number of pathlines; for example, a typical computation of the FTLE computes pathlines for every grid point in the domain. Furthermore, these pathlines are computed using numerical integration such as Runge-Kutta (RK) schemes, and require interpolation as well as integration of vectors in both space and time. As a result, they lack numerical robustness, and are affected by errors and uncertainties that are often nontrivial to track. A lot of effort has been invested into making the pathline computation faster and numerically more robust, especially since the introduction of the FTLE to the visualization community in the year 2000 [80].

Yet, there exist more challenges—both theoretical and computational—in using pathlines and the FTLE for unsteady flow analysis. Recently, Haller [83] showed that the usual way of computing material boundaries as the ridges of the FTLE field is not reliable, as particles can cross these ridges. Furthermore, the FTLE must trace pathlines for a finite amount of time, requiring simultaneous access to data across time-steps. Flow data is typically available in Eulerian form, where vector values are assigned to known grid points. This Eulerian data is stored as snapshots for each available time-step. Therefore, the FTLE and the other pathline-based approaches require loading data from individual

time-steps into the memory at high frequency, making the computation inefficient. The lack of scalability and efficiency of such techniques make them unattractive for large-scale data. Especially considering the future of data analysis, where efficient and scalable techniques will be required to work at exascale, the approaches based on pathlines are not promising due to their lack of scalability, despite the parallelization they offer.

Nevertheless, these limitations are often disregarded, and pathline-based approaches are still used to analyze unsteady flows, because the underlying theoretical premise is that they capture particle behavior, and as a result, produce temporally coherent and physically meaningful features. The next section presents a case study to demonstrate that, unfortunately, the pathlines computed from the sampled flow can be drastically different from the actual particle behavior, especially for large-scale simulated data.

5.1 A Case Study in Large-scale Turbulent Combustion

Using a large-scale turbulent combustion simulation as a case study, this section evaluates a set of pathlines computed using the state-of-the-art pathline tracers in postprocessing, against a corresponding set of highly accurate pathlines computed directly alongside the simulation. Considering the simulation under examination to be emblematic of a large class of simulations, it is important to note that the available temporal resolution of the data is strictly limited by the number of time-steps stored by the simulation, which are typically severely restricted by the file I/O rates and the amount of disk space available, which raises concerns that the simulations might be saving too little data to accurately compute pathlines. Unfortunately, for most practical applications, evaluating these errors is difficult as scientists almost universally will stress their available resources to the limit, and thus storing substantially more data is infeasible. In the absence of better solutions, pathline-based techniques are simply applied to the best data available with the implicit assumption that the temporal resolution is sufficient to reliably analyze the underlying scientific phenomena. The evaluation presented here demonstrates that, unfortunately, this is not true. In particular, it will be shown, both qualitatively and quantitatively, that there can exist differences substantial enough that any constructs built upon such computed pathlines, such as the FTLE, do not necessarily capture the flow behavior, making the conventional pathline-based analysis and visualization prone to large errors, rendering their results unreliable at best and possibly misleading at worst.

5.1.1 Experimental Setup

This case study uses a large-scale turbulent combustion simulation of the lifted ethylene jet flame [222, 223] performed using S3D [29]. S3D is designed to perform massively parallel direct numerical simulation for turbulent combustion, which uses a Runge-Kutta integrator to solve the full compressible Navier-Stokes equation, and balance equations for conserving mass, momentum, energy, and chemically reactive species. The resulting flow (see Data B.3.1) is used to investigate turbulent lifted jet flames with the goal of improving the understanding of direct injection stratified spark ignition engines for commercial boilers, as well as fundamental combustion phenomena. A visualization of a 2D slice of the flow field through the center of the frame can be found in Appendix B.

The flow under consideration is defined on a $[2025 \times 1600 \times 400]$ rectilinear grid, with a variable spacing in the y dimension. The data was generated on Jaguar (now Titan) at the Oak Ridge leadership computing facility, using $30,000$ cores with $[27 \times 40 \times 40]$ grid points per processor. The simulation captures a large number of variables representing the flow velocity, temperature, pressure, species, etc., and the total file-size for storing these variables, just for a single time-step, is about 280 GB.

Due to large I/O overheads, the storage and management of such huge amounts of data becomes infeasible. As a result, such simulations typically save data snapshots only every 500^{th}–1000^{th} time-step. In this particular case, data has been saved at an unusually high frequency to study ignition phenomena ahead of the flame base. According to the domain experts who generated this data, it is unlikely that any similar combustion simulation would save data more frequently, making the results discussed below even more poignant. Even though most quantities of interest do not change significantly over a timescale of 10 to 100 time-steps [224], the standard temporal resolution has been shown to be too low for reliable tracking of important features, such as regions of locally high scalar dissipation [136]. Furthermore, as is the case with most numerical simulations, S3D employs high levels of accuracy, for example, an eighth-order approximation to compute the spatial derivatives on a finite-difference grid. However, due to the cutbacks in temporal resolution, this high level of accuracy is not translated into the postprocessing analysis and visualization stage.

5.1.1.1 In situ particle tracing. This simulation of the lifted flame, for the first time, also included a collection of Lagrangian particles for a part of the run. In particular, a total of $1,248,550$ particles were tagged, and advected directly alongside the simulation at a significantly higher frequency. These particles are stored for 299 time-steps uniformly distributed in the time range $[1.7, 1.7598] \times 10^{-3}$ with a temporal step-size of 2×10^{-7}.

Therefore, these particles are available at an unusually high temporal resolution as compared to the full snapshots, and effectively represent a set of highly accurate pathlines.

All the other simulation variables were saved only for 30 time-steps in the considered time-range with a temporal step-size of 2×10^{-6}, a factor of 10 slower. Note that, even at a tenth of the resolution of the in situ particles, storing the data associated with the simulated variables for this short time-range requires more than 8 TB, thus highlighting the challenges in increasing the temporal resolution of the saved data. Normally, the ratio of full snapshots saved to simulation time-steps is one or two orders of magnitude worse.

5.1.1.2 Postprocessed pathline computation. We implemented a custom integrator for pathlines computation. Using the fourth-order Runge-Kutta (RK4) integrator, our pathline computation allows configuring the step-size in terms of the number of integration steps to be performed between two consecutive time-steps of the flow. Working on rectilinear grids, the integrator uses trilinear interpolation in the 3D space and linear interpolation in time. To overcome memory usage issues, we converted the flow data into IDX format [158, 159]. The IDX file format reorganizes the data based on multiresolution space-filling curves and provides a simple API for flexible and efficient loading of blocks of data without need of alignment with the chunks of the original computation. Our custom pathline computation takes advantage of the IDX format by loading only those blocks of data that are required by the currently progressing pathlines.

We utilize shared memory multithreading to parallelize our pathline tracer. Each thread from a pool of 16 threads, in a loop, picks a pathline to perform integration for a single time-step. The entire flow volume is logically divided into blocks of size 64^3, and only the blocks that are required by currently progressing pathlines are loaded, and shared among the threads. The blocks are cached in memory and are freed in a least-recently-used manner, when the memory usage crosses a threshold (4 GB). Integration for a single time-step at a time provides temporal locality in computation, and by restricting the amount of time-steps to be loaded, it reduces the required disk I/O.

We also implemented parallel pathline tracing using the `vtkPParticlePathFilter` class of the visualization toolkit (VTK) [185], with the RK4 integrator. The VTK is an open-source, cross-platform software that provides libraries for a variety of visualization tasks. The filter performs two integration steps between two consecutive time-steps of the data. However, we found the VTK implementation of pathline tracing to be rather inefficient, because, for the purpose of parallelization, it completely relies on message passing instead of utilizing shared memory or a hybrid of both, preventing any sharing of data. Furthermore,

the VTK implementation parallelizes the computation over blocks of volume data, and loads the entire volume for each time-step, even if the pathlines are confined to a smaller region. Nevertheless, we used the VTK implementation to validate our custom pathline integrator, which we used for all the experiments presented next.

5.1.2 Comparing Pathlines

Given two discrete pathlines $\mathbf{p}[t]$ and $\mathbf{q}[t]$ of the same particle computed through different sources, the distance between them can be measured in terms of their average point-wise distance, that is,

$$\varepsilon(\mathbf{p}, \mathbf{q}) = \frac{\sum_{i=0}^{n-1} (\mathbf{p}[t_i] - \mathbf{q}[t_i])}{n},$$

where, n is the length of the pathlines, and ε is the error vector. Since the locations of the in situ particles are given in the grid coordinates, the computed pathlines must be converted from physical space to grid space. Thus, each unit of distance represents a distance of one grid cell. This evaluation focuses on a smaller region around the flame, which was chosen by the domain scientists to be the most interesting and most turbulent part of the flow. This region contains $54,935$ particles, for which the comparison between the postprocess pathlines and the in situ paths of these particles can be performed.

Since the flow data is available for 30 time-steps only, as compared to the 299 time-steps for which the location of in situ particles is known, we use the locations of the in situ particles for every tenth time-step to ensure that the point-wise locations are used for the same time values, leading to $n = 30$. However, in certain cases, either of the pathlines may exit the domain while the other still lies inside, thus giving fewer than 30 points to compare. To handle such cases, we take n to be the length of the shorter pathline, meaning that the comparison is terminated as soon as any of the pathlines exits the domain. Finally, the in situ particles wrap around in the z dimension, whereas in our pathline computation, we terminate the tracing when the pathline goes out of bounds to disallow wrapping of pathlines. Such cases are also handled by taking n to be the length of the shorter of the two pathlines.

In order to study the comparison, histograms of the components of the distance vector ε in Cartesian coordinates $\{\varepsilon_x, \varepsilon_y, \varepsilon_z\}$ have been used. By highlighting the distances in the three dimensions separately, these histograms illustrate the influence of the flow on pathlines in the corresponding dimensions. Histograms of ε in spherical coordinates $\{\varepsilon_\rho, \varepsilon_\theta, \varepsilon_\phi\}$ are also useful in studying the comparison. Here ρ is the radial distance, θ represents the azimuthal angle, that is, $\theta = \tan^{-1}(y/x) \in [-180°, 180°]$, and ϕ represents the polar angle,

that is, $\phi = \cos^{-1}(z/\rho) \in [0°, 180°]$. The histograms in the spherical coordinates illustrate the spread of error in the xy plane, as well as with respect to the inclination above and below this plane.

5.1.2.1 Validating offline-computed pathlines. The first result presented here validates our custom particle integrator by comparing it with the standard VTK implementation. Both implementations use RK4 integrator with two integration steps between consecutive time-steps. As shown by the histogram in Figure 5.1, the two implementations produce very similar pathlines, where most of the differences exist due to numerical details of the implementations and the known instabilities in turbulent flows.

5.1.2.2 Offline-computed pathlines versus in situ particles. A visual comparison between the pathlines computed using our custom implementation with the in situ particles is shown in Figure 5.2, and already shows substantial differences in the two sets of pathlines. The error vector ε is computed by subtracting the location of in situ particles from the corresponding locations in computed pathlines. Figure 5.3 shows the histograms of this comparison, where 1000 integration steps have been used between two consecutive time-steps of the flow. The results illustrate that even for a very small integration step $(\approx 10^{-9})$, the available temporal resolution of the data is not sufficient to capture the true behavior of the particles. In particular, the variance of the error distribution of the distances shows that most pathlines differ from the corresponding in situ particles by about 16 grid cells, and a substantial number of pathlines deviate by up to 50 grid cells. The error distributions in the x, y, and z dimensions (see Figure 5.3 top) show that the error is highest in the x dimension, and lowest in the z dimension. This observation matches the understanding of the flow since the flow is fastest in the x dimension and any small errors

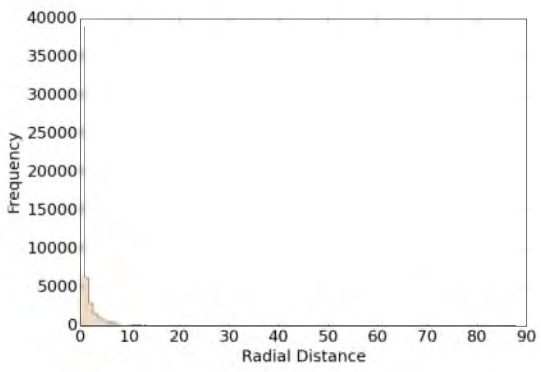


Figure 5.1. Mean distances between pathlines computed through our custom integrator and the VTK integration are negligible, validating the former.

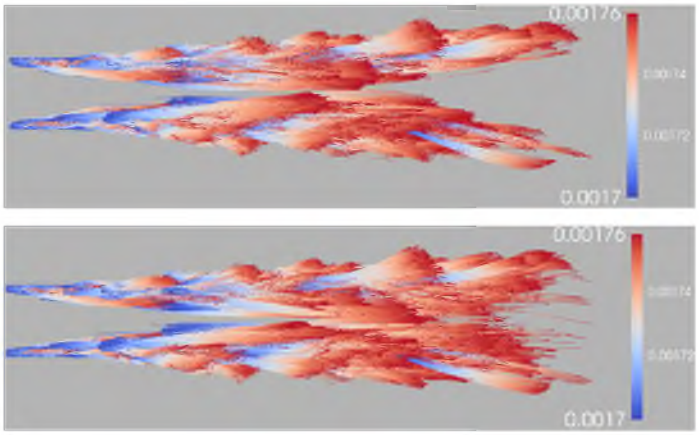


Figure 5.2. A visual comparison of in situ (left) and offline computed (right) pathlines illustrates that the latter do not capture the true particle behavior. The color map on the pathlines represents the time-range $[1.7, 1.7598] \times 10^{-3}$.

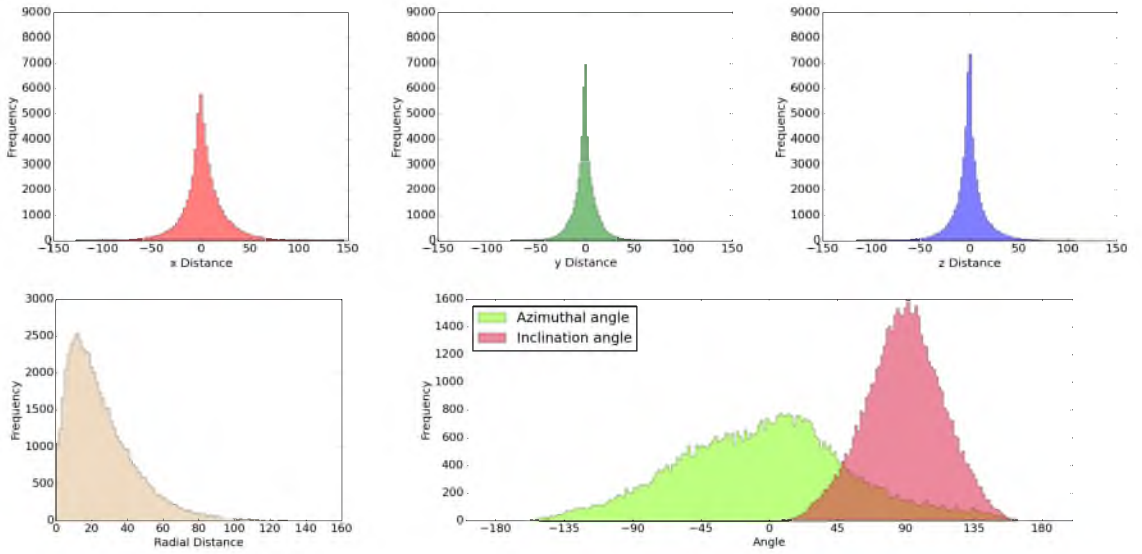


Figure 5.3. Histograms of the mean distances between in situ particles and computed pathlines using all 30 available time-steps covering the time-range $[1.7, 1.7598] \times 10^{-3}$. The histograms of error distribution in Cartesian (top) and spherical (bottom) coordinates show that the x dimension incurs the highest error rate, and highlights the high variance in the error distribution of the azimuthal angle.

are propagated at fast speeds, whereas the slow magnitudes in the z dimension restrict the corresponding propagation of errors.

The histograms in spherical coordinates shed more light on the nature of error. Figure 5.3 shows that the variance of the azimuthal angle is very high, and that most of the computed pathlines were "ahead of" (faster than) the in situ particles, that is, $|\theta| < 90°$. The speculation is that the distance between the speeds exists because postprocessing pathlines encounter less turbulence in general—an artifact of the simulation's inability to translate higher-order approximations to the postprocessing phase. On the other hand, some pathlines lagged "behind" (slower than) the in situ particles, that is, $|\theta| > 90°$, highlighting the randomness in the error vector within the xy plane.

The randomness in the error is also corroborated by Figure 5.4, which shows the spatial distribution of the errors in spherical coordinates by color-mapping them to the seed points of the pathlines. This lack of a characteristic nature in the error has serious impacts on any subsequent analysis, for example, in using the FTLE field to compute the Lagrangian coherent structures (LCS) of the flow. The FTLE field measures the separation between neighboring pathlines after a finite amount of time. However, due to the dynamic nature of the flow, the randomness in the error during pathline computation can propagate the

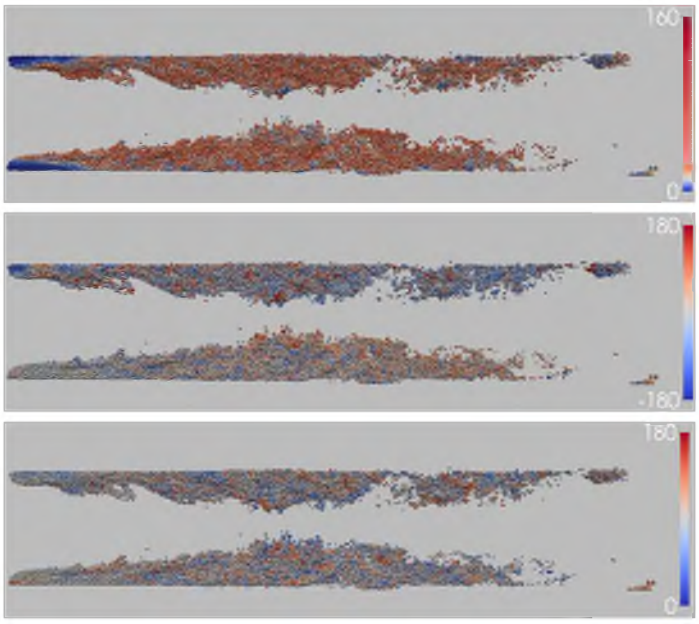


Figure 5.4. Spatial distribution of the mean distances between in situ particles and computed pathlines. The seed points of pathlines are colored based on the mean radial distance (top), mean azimuthal angle (middle), and mean polar angle (bottom). The figure shows that the spatial distributions of these errors are unorderly, potentially leading to arbitrary errors in any subsequent analysis.

particles to completely different locations, resulting in an unbounded error in the FTLE field. For example, distances between two infinitesimally close pathlines lying on either side of a material boundary can increase considerably even due to a small error in computation. A similar example in the context of streamlines in steady flows is given in Chapter 6. Compounded by the randomness in the nature of the error, the ultimate analysis becomes unpredictable and unreliable.

5.1.2.3 Comparing different lengths of pathlines. For the same settings of the integrator, Figures 5.5 and 5.6 show the comparisons between the pathlines for the first 10 time-steps, covering the time-range $[1.7, 1.7198] \times 10^{-3}$, and the first 20 time-steps covering the time-range $[1.7, 1.7398] \times 10^{-3}$ (as compared to the total of 30 time-steps covering the time-range $[1.7, 1.7598] \times 10^{-3}$). These results convey how the error increases as the pathlines are allowed to propagate for longer times. Note that whereas the errors in the x, y, and z dimensions, and also in the radial distance increase, in terms of azimuthal and polar angles, they remain similar to the original case (see Figure 5.3) reiterating the randomness in its distribution, as discussed above.

5.1.2.4 Comparing different temporal resolutions. The final comparison is that of the pathline computation using every second available time-step to understand

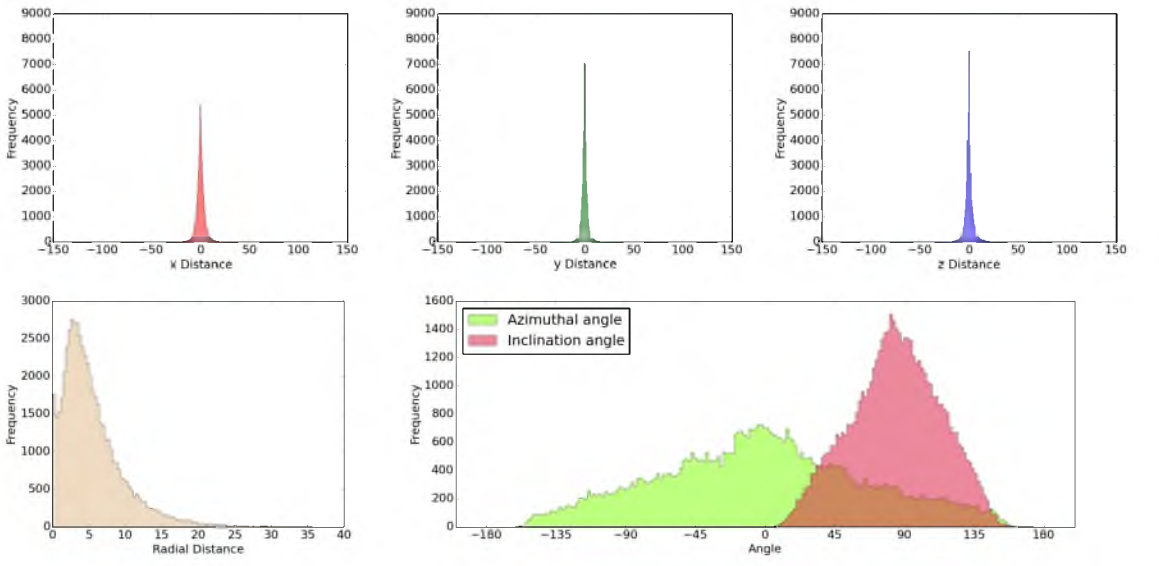


Figure 5.5. Histograms of the mean distances between in situ particles and computed pathlines for the first 10 time-steps covering the time-range $[1.7, 1.7198] \times 10^{-3}$. The histograms of error distribution in Cartesian (top) and spherical (bottom) coordinates illustrate that the error is lower in general (as compared to results in Figures 5.6 and 5.3) due to the smaller time-range. Note that the error distribution in the azimuthal angle has a high variance.

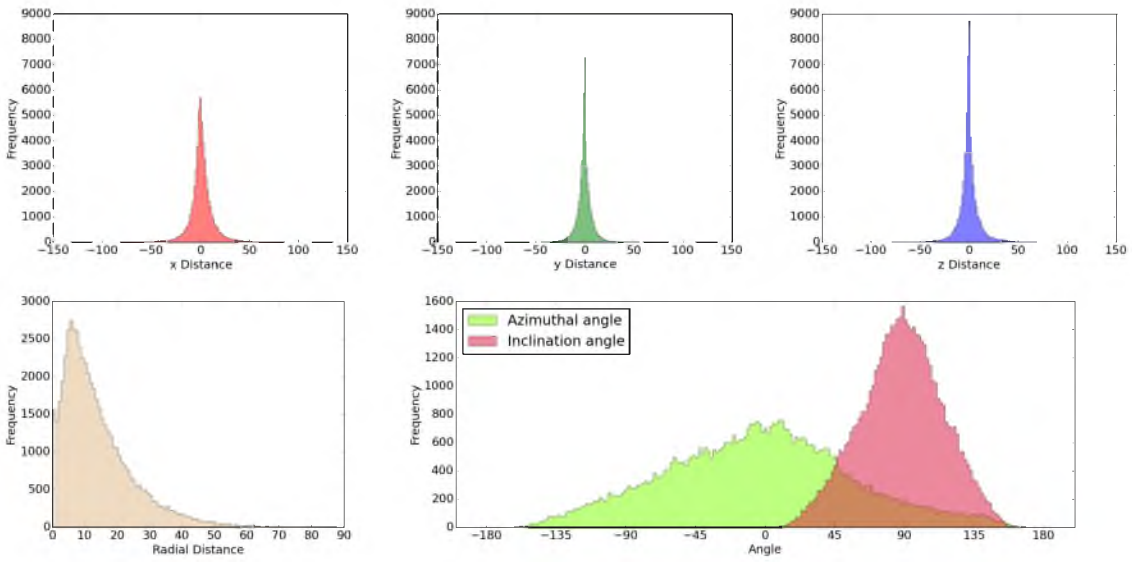


Figure 5.6. Histograms of the mean distances between in situ particles and computed pathlines for the first 20 time-steps covering the time-range $[1.7, 1.7398] \times 10^{-3}$. The histograms of error distribution in Cartesian (top) and spherical (bottom) coordinates show that the errors increase when pathlines are propagated for a longer time-range. Compare these results with the results when only the first 10 (Figure 5.5), and all 30 time-steps (Figure 5.3) are used.

the impact of temporal resolution on pathline computation. Thus, data from a total of 15 time-steps, covering the same time range $[1.7, 1.7598] \times 10^{-3}$, was used, and the resulting pathlines were compared with the in situ particles (every 20^{th} particle location).

Figure 5.7 shows the result for this comparison for pathlines computed at half the available resolution. As compared to Figure 5.3, which shows the results computed using the highest available temporal resolution, the plots in Figure 5.7 are not significantly worse. In particular, the peak of the error distribution of the radial distance increases from 16 to 20 grid cells. These comparisons suggest that doubling the temporal frequency improves the accuracy only by a small amount. Judging by the overall distribution of error in Figure 5.3, it appears that reducing the error to acceptable levels for reliable analysis will require substantial scaling up of temporal resolutions. The implication is that the analysis performed at the highest available temporal resolutions is already not producing meaningful results due to large amounts of errors.

5.2 Summary

Almost all modern large-scale simulations are capable of simulating physical phenomena in a highly-accurate manner. Unfortunately, this accuracy is typically lost since the resulting

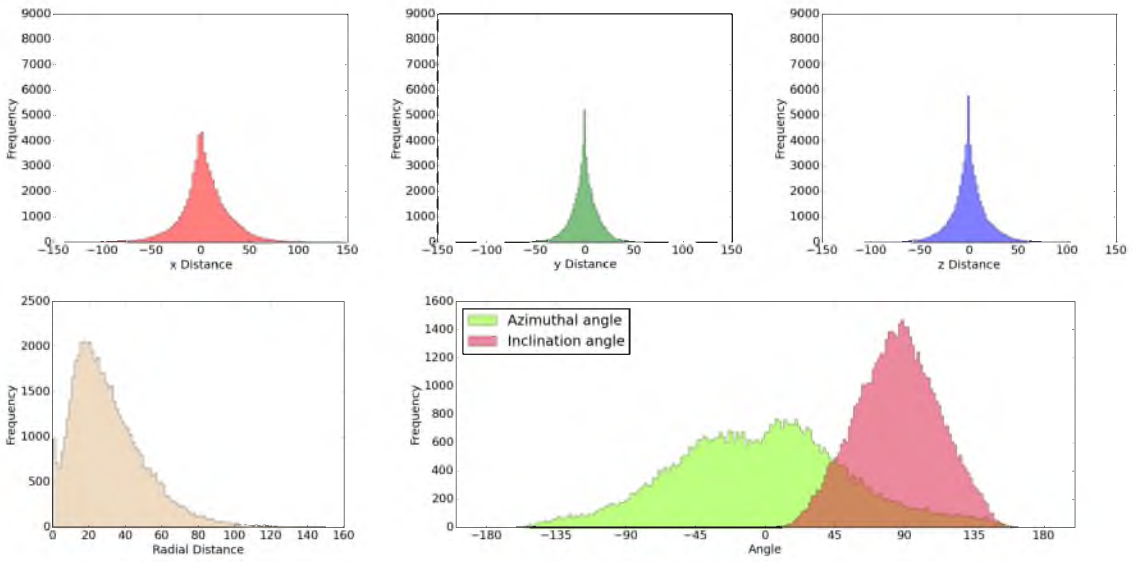


Figure 5.7. Histograms of the mean distances between in situ particles and computed pathlines using every second available time-step, that is, using only 15 time-steps covering the time-range $[1.7, 1.7598] \times 10^{-3}$. The histograms of error distribution in Cartesian (top) and spherical (bottom row) coordinates show that the growth of error is not substantial, suggesting that a much higher temporal resolution is required for reliable analysis.

data can be stored only at much lower temporal resolutions. To analyze and visualize this data, offline techniques make an implicit assumption that the available temporal resolution is sufficient to obtain reliable and physically meaningful results. Using a case study from a large-scale combustion simulation, this chapter questions this assumption, and shows that the offline computed pathlines deviate substantially from the actual particle behavior in the flow. These results suggest a lack of reliability in the common pathline-based approaches.

Although this evaluation focused on a single case study, a large variety of turbulent flows can be expected to suffer from the same problems due to the complexity of the underlying phenomena as well as the I/O bottlenecks. In the opinion of the domain experts, few, if any, combustion simulations would store data even at the given frequency. Since large-scale simulations in all areas operate under the same "as little as possible, as much as necessary" paradigm when it comes to checkpointing, these results raise significant, far-reaching concerns regarding the reliability of pathline-based techniques.

This first-of-its-kind study raises new questions regarding the applicability of postprocessing techniques based on pathlines, stressing the need for tighter coupling of simulations and analytics using new and alternative perspectives in large-scale unsteady flow analysis. In particular, since the data available within the simulation is rich of highly accurate

information, in situ techniques appear to be very promising. However, such techniques would have access to single time-steps only.

Nevertheless, despite the availability of much simpler, robust, and well-understood steady flow analysis techniques, in practice, they are underutilized, because they cannot be, in general, applied to time-varying data, which is used to represent almost all flows of practical interest. To this end, Chapter 4 of this dissertation develops new reference frames, and discusses in detail the significance of these reference frames in unsteady flow analysis. In particular, these reference frames make the steady flow analysis techniques applicable to individual time-steps of unsteady flows.

PART III

COMBINATORIAL REPRESENTATIONS

CHAPTER 6

REPRESENTATION AND ANALYSIS OF FLOW WITH BOUNDED ERROR

Typically, vector fields are stored as sets of sample vectors at discrete locations. Both a discretization of the domain of the field, often in the form of a structured or unstructured mesh, as well as a set of sampled vectors at the vertices of the mesh, are defined. The vector field on the interior of the mesh is approximated by interpolating vector values from these samples at the mesh's vertices. The simplest interpolation scheme constructs a linear field along the edges joining two vertices, leading to, for example, a bilinear and trilinear interpolation in the case of structured rectangular meshes in 2D and 3D, respectively. One of the most common forms of 2D meshes is triangulated in nature, where such a linear interpolation using the three vertices of the triangle is also called barycentric interpolation, and constructs a piecewise linear (PL) representation of the flow. Subsequently, computing flow properties that require integrating these vector values presents a significant computational challenge.

For example, consider the computation of streamlines, which, in the case of steady flows, represent the paths of massless particles that travel using the instantaneous velocity defined by the flow. They are the most fundamental construct of flow analysis—to the extent that they are sometimes directly referred to as *flow*. Streamlines are used to construct the topological skeleton of the flow [89], visualization of the flow [20, 208], etc. The computation of streamlines is typically performed using numerical integration, such as the Euler integration or the Runge-Kutta (RK)-based techniques [170]. However, using such numerical techniques to compute streamlines poses significant challenges. In particular, these techniques take a small step in the current direction of the vector and recompute the new direction using an interpolation scheme to repeat the process. For example, using the Euler integration (illustrated in Figure 6.1), the position of the particle at $(n+1)^{\text{th}}$ step is calculated as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \hbar\, \vec{V}(\mathbf{x}_n), \tag{6.1}$$

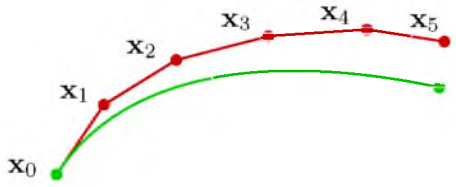


Figure 6.1. Numerical integration error in streamline tracing. Small errors are incurred at each integration step, which can accumulate and cause the numerical streamline (red) to deviate substantially from the true streamline (green).

where $\hbar$ denotes the step-size and $\vec{V}(\mathbf{x}_n)$ represents the vector at the n^{th} position. Each integration step introduces some error, which is typically known as an upper bound with respect to the step-size. During the integration, however, these small errors get compounded, and it becomes nontrivial to determine the net error incurred during the streamline tracing. Despite these problems, many of the standard analysis techniques used for vector fields rely on variants of RK methods. However, robust computation of flow becomes a formidable task, since the errors in numerical integration schemes can compound quickly, resulting in inaccurate visualization and analysis of the field.

This chapter introduces *Edge Maps*, which provide an explicit representation of flow by mapping entry and exit points of flow paths on the edges of the triangle. Thus, they encode the property most often needed by common analysis tools to compute visualizations and topological decompositions. As a result, many of the same primitives can be computed robustly and directly on the Edge Maps themselves. Moreover, the Edge Map data structure encodes numerical error, presenting a complete view of the data illuminating the major features that demonstrate where numerically unstable regions exist. Consequently, instead of providing a black-box representation of the data that ignores the impact of discretization, Edge Maps provide analysts a visualization of the data that accounts for these artifacts, and indicate how errors may have affected the apparent flow behavior. The encoding of error also enables refinement of the maps to bound the amount of error incurred by this representation.

The focus of this chapter is to develop an understanding of the properties of Edge Maps, whereas Chapter 7 will revisit the notion of consistency, and provide more details on how to obtain consistency guarantees in streamline tracing using Edge Maps.

6.1 Properties of Piecewise Linear (PL) Flow

Let $\vec{V}: \mathbb{M} \to \mathbb{TM}$ be a tangential sampled vector field defined on a 2-manifold $\mathbb{M}$ embedded in $\mathbb{R}^3$ (see Definition 2.8). Typically, the space $\mathbb{TM}$ is represented as a triangulation

$\mathcal{M} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}\}$, where $\mathcal{V}$, $\mathcal{E}$, and $\mathcal{T}$ are sets of vertices, edges, and triangles, respectively, in the triangulation. Then, $\vec{V}$ is represented as the set of vector values sampled at the vertices $\mathcal{V}$ of $\mathcal{M}$, that is, $\vec{V}(\mathbf{v}_i)$ for all $\mathbf{v}_i \in \mathcal{V}$. The vector values at the interior of each triangle Δ with vertices $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\}$ in the triangulation are interpolated linearly using $\vec{V}(\mathbf{v}_i), \vec{V}(\mathbf{v}_j)$, and $\vec{V}(\mathbf{v}_k)$[1]. Figure 6.2 depicts the field defined in this way for a single triangle. Such a definition computes a piecewise linear (PL) vector field, which is linear within each triangle, and continuous across the edges of the triangulation.

The remainder of this work assumes that the given vector field $\vec{V}$ is generic (see Definition 2.10), meaning that the vectors at all the vertices of the triangle are nonzero. This work further makes the following two assumptions.

Assumption 6.1. The vectors at any two vertices sharing an edge are not antiparallel.

Assumption 6.2. The vectors at two vertices on an edge are not both parallel to that edge.

Any configuration that violates these assumptions is unstable, and can be avoided by a slight perturbation. In particular, the Simulation of Simplicity (SoS) [49] (see Section 2.4.3) can be used to satisfy these assumptions using symbolic perturbations. These perturbations ensure a point-to-point mapping between the boundary of the triangle, thus making Edge Maps bijective. As will become clear in the following sections, these assumptions significantly reduce the complexity of many of the properties of Edge Maps, because the locations of critical points become well defined.

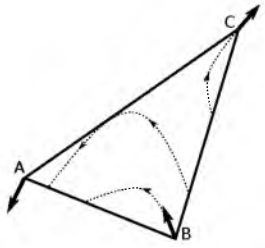


Figure 6.2. Traditional way of representing vector fields store vectors at the three vertices of the triangle, which complete the flow at the interior using interpolation.

[1]Note that the three vectors must reside in the plane of the corresponding triangle. In the case of nonplanar domains, the projection of a given vector $\vec{V}(\mathbf{v}_i)$ is different for each triangle sharing the vertex $\mathbf{v}_i$. To avoid this inconsistent representation, mean value coordinates in 3D [53] can be used to project the star of $\mathbf{v}_i$, $\mathrm{St}(\mathbf{v}_i)$, onto a 2D plane. The vector $\vec{V}(\mathbf{v}_i)$ can then be projected onto every triangle in $\mathrm{St}(\mathbf{v}_i)$, and an average vector can be computed in 2D. The inverse projection (to 3D) of the average vector can then be computed for every triangle in $\mathrm{St}(\mathbf{v}_i)$ giving a consistent representation of $\vec{V}(\mathbf{v}_i)$.

Under the aforementioned assumptions, the properties of linearly varying vector fields within each triangle can be studied. Since this chapter focuses on steady vector fields, it will loosely use the term *flow* for the streamlines of the field, which describe the parametric paths of massless particles in steady vector fields. Flow is an important property that will be useful in understanding the representation presented in this chapter. Two important properties that directly follow from the aforementioned assumptions are given below.

Property 6.1. Within Δ, $\vec{V}$ defines at most one critical point.

Proof. Critical points are defined as points $\mathbf{x}$ where $\vec{V}(\mathbf{x}) = \vec{0}$. Consider the component scalar fields u and v, where $\vec{V}(\mathbf{x}) = (u(\mathbf{x}), v(\mathbf{x}))$. Since $\vec{V}$ and, hence, both u and v are linearly varying, the graphs of u and v are planes lifted from the triangle. At a critical point, the planes of both u and v intersect the constant plane of height zero. From planar geometry, it is known that two planes either intersect in another plane or a line, and consequently these three planes intersect in a plane, a line, or a point.

The intersection of these three planes can be a plane, in which case, there exist infinitely many zeros, meaning that the entire vector field is zero (not generic). If these planes intersect in a line, the only way that line passes through Δ is when either the field is not generic (because two vertices are zero) or assumption (1) is violated for at least two edges. In all other cases, either a single critical point is defined (within the interior of Δ) or the field defines critical values outside of Δ. Therefore, a linear vector field satisfying the said assumptions defines at most one critical point within Δ. □

Property 6.2. On the line ℓ obtained by extending, from both sides, any edge e of a triangle Δ, there exists at the most one point where the vector field is tangential to ℓ.

Proof. Without loss of generality, rotate ℓ to be the x axis. Consider the component scalar fields u and v, where $\vec{V}(\mathbf{x}) = (u(\mathbf{x}), v(\mathbf{x}))$. Following the linearity of the vector field $\vec{V}$, v is also linear. By assumptions (3) for a generic field, $v = 0$ can happen only once, meaning that $\vec{V}$ is tangential to ℓ at only one point. Furthermore, on either side of this zero crossing, v has a different sign, and hence the flow changes directions from flowing upwards to downwards, or vice versa. □

6.2 Edge Maps: New Representation of PL Flows

With an understanding of some important properties of PL vector fields, the basic elements of a new representation for vector fields, called *Edge Maps*, are discussed next.

6.2.1 Fundamental Elements of Edge Maps

For a triangle Δ, let $\partial\Delta$ and $\mathring{\Delta}$ denote the boundary and interior of Δ, respectively. We first examine the behavior of the flow at any point on $\partial\Delta$. The behavior of a particle at the boundary of a triangle can be classified into four types (see Figure 6.3) depending upon the location of that particle under the flow for positive and negative time t. Given a point $\mathbf{p} \in \partial\Delta$ such that $\mathbf{x}(0) = \mathbf{p}$, and for $\epsilon > 0$, we define the following:

Definition 6.1 (Inflow Point)**.** If for all $t \in]0, \epsilon]$, $\mathbf{x}(t) \in \mathring{\Delta}$ and $\mathbf{x}(-t) \notin \Delta$, then $\mathbf{p}$ is an *inflow point.*

Definition 6.2 (Outflow Point)**.** If for all $t \in]0, \epsilon]$, $\mathbf{x}(t) \notin \Delta$ and $\mathbf{x}(-t) \in \mathring{\Delta}$, then $\mathbf{p}$ is an *outflow point.*

Definition 6.3 (External Transition Point (ETP))**.** If for all $t \in]0, \epsilon]$, $\mathbf{x}(\pm t) \notin \Delta$, then $\mathbf{p}$ is an *external transition point (ETP).*

Definition 6.4 (Internal Transition Point (ITP))**.** If for all $t \in]0, \epsilon]$, $\mathbf{x}(\pm t) \in \mathring{\Delta}$, then $\mathbf{p}$ is an *internal transition point (ITP).*

Intuitively, the inflow and outflow points are the points on the boundary of a triangle where the flow enters and exits the triangle, whereas the transition points are the ones where the flow is instantaneously tangential to the boundary, but remains either outside (for an ETP) or inside (for an ITP) the triangle. The concept of transition points is similar to that of boundary switch points [39, 125, 213]. The above four definitions account for all possible behaviors of flow as it touches the boundary of the triangle, since the two assumptions presented in the previous section disallow the possibility of particles traveling along the edge. To model the flow through the interior of the triangle, we start with pairing these boundary points if they exist along the same streamline.

Definition 6.5 (Origin-Destination Pair (OD pair))**.** Let $a, b \in \mathbb{R}$ and $a \leq b$ such that

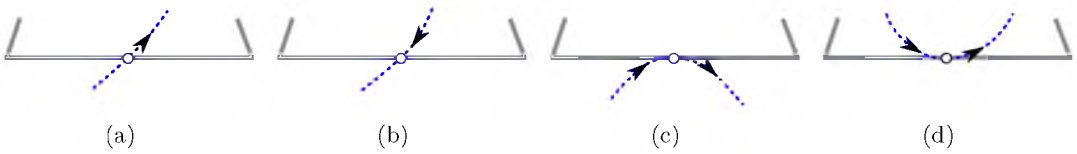


Figure 6.3. Classification of points on the boundary of a triangle based on the flow. (a) Inflow point; (b) Outflow point; (c) External transition point (ETP); and (d) Internal transition point (ITP).

$\mathbf{x}([a,b]) \subset \Delta$, where $\mathbf{p} = \mathbf{x}(a)$ and $\mathbf{q} = \mathbf{x}(b)$. Then, $(\mathbf{p}, \mathbf{q})$ is called an *origin-destination (OD) pair*, if the time interval $[a,b]$ is maximal where $\mathbf{p}, \mathbf{q} \in \partial\Delta$ and $\mathbf{x}(]a,b[) \subset \mathring{\Delta}$. The point $\mathbf{p}$ is called an *origin point*, and $\mathbf{q}$ a *destination point*.

Here, maximal interval means that the time range $[a,b]$ produces the largest possible streamline contained within $\mathring{\Delta}$, bounded by two points $\mathbf{p}$ and $\mathbf{q}$ on $\partial\Delta$. Inflow, outflow, and transition points all play different roles with respect to OD pairs. The simplest case is for inflow and outflow points. Since the streamline of an inflow point $\mathbf{p}$ flows to the $\mathring{\Delta}$, $\mathbf{p}$ will be paired with the point $\mathbf{q}$ (either an outflow point or ITP) where this streamline first touches the boundary after $\mathbf{p}$.

An ITP flows to $\mathring{\Delta}$ in both positive and negative time, thus creating at most two such maximal intervals for which a streamline is completely contained in $\mathring{\Delta}$. Hence, an ITP may participate in two OD pairs. In one pair, it is an origin point, whereas in the other, it is a destination point. If the streamline is an orbit touching $\partial\Delta$ only at the ITP, the ITP is paired with itself, resulting in a single OD pair. In this case, the streamline of the orbit has many time intervals $[a,b]$ where $\mathbf{x}(]a,b[) \subset \mathring{\Delta}$, all of which are of equal length. On the contrary, an ETP always forms an OD pair with itself, since the streamline does not flow to $\mathring{\Delta}$. Hence, its maximal time range will be when $a = b$.

Thus, all origin points are either inflow points or transition points, and all destination points are either outflow points or transition points. Note that if $\mathbf{x}$ is chosen such that $\mathbf{x}(t)$ is an orbit contained entirely in $\mathring{\Delta}$, then such a streamline will never converge to $\partial\Delta$. Hence no points on such a streamline will form an OD pair. Moreover, there are some points that do not converge to $\partial\Delta$, because of the presence of a critical point in Δ. Such points can be classified as *unmapped inflow points* (associated with a sink), *unmapped outflow points* (associated with a source), or *sepx points* (intersections of the saddle separatrices with $\partial\Delta$), and they are not included in the definition of OD pairs.

Using these basic elements, Edge Maps for *regular triangles*—those that do not contain a critical point at their interior—are defined next. After that, Section 6.2.4 will extend these definitions to generalize Edge Maps triangles that may contain critical points.

6.2.2 Edge Maps

Since the existence of an OD pair for every point on $\partial\Delta$ is deterministic, a mapping can be defined to describe the transversal behavior of flow through points on $\partial\Delta$. Let $\mathbf{P} \subset \partial\Delta$ be the set of all origin points, and $\mathbf{Q} \subset \partial\Delta$ be the set of all destination points.

Definition 6.6 (Edge Map). An *Edge Map* of triangle Δ is defined as a map $\xi\colon \mathbf{P} \to \mathbf{Q}$, such that $\xi(\mathbf{p}) = \mathbf{q}$, if and only if $(\mathbf{p}, \mathbf{q})$ is an OD pair. Then, $\mathbf{q}$ will be called the *image* of $\mathbf{p}$ under ξ.

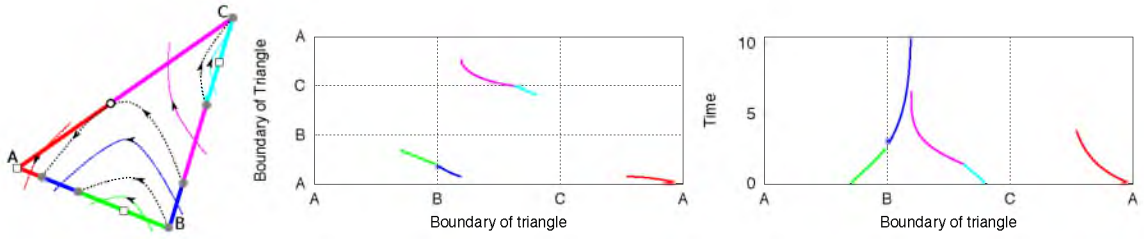
Whereas some flow constructs such as topological segmentation are invariant to the vector magnitudes, others such as streamlines are not. To represent the flow accurately, it is important that the speed of the flow is encoded into Edge Maps as well.

Definition 6.7 (Time-Edge Map). A *Time-Edge Map* of a triangle Δ is defined as a map $\xi_t\colon \mathbf{P} \to \mathbb{R}$, such that $\xi_t(\mathbf{p})$ describes the time it takes the flow to reach from $\mathbf{p}$ to its image, $\mathbf{q} = \xi(\mathbf{p})$.

An illustration of Edge Map and Time-Edge Map is given in Figure 6.4. Considering triangles without critical points, it can be noted that the flow within them is a bijection between $\mathbf{P}$ and $\mathbf{Q}$, and thus its inverse is also well defined. As a result, whereas an Edge Map, ξ, maps an origin point to its destination point under forward flow, its inverse ξ^{-1} maps a destination point to its origin point under backward flow.

6.2.3 Approximations of Edge Maps

For a practical implementation, a feasible representation for storing and using Edge Maps (and Time-Edge Maps) is needed. This is achieved by grouping the point-based, exact mapping ξ, into connected intervals that preserve its ordering, creating *links*.



(a) (b) (c)

Figure 6.4. Edge Map for the triangle from Figure 6.2. (a) Edge Map subdivides the boundary into a set of links, which map inflow to outflow for a triangle. (b) Edge Map visualized as a plot, mapping the origin points (horizontal axis) to destination points (vertical axis). Each point on the plot represents a streamline between an OD pair. Since Edge Map is defined only for origin points on the boundary, the plot is disconnected at the points where the boundary switches from inflow to outflow, or vice versa. (c) Time-Edge Map visualized as a plot, mapping the origin points (horizontal axis) to the time taken by them to reach their respective destinations (vertical axis).

Definition 6.8 (Link)**.** Let $\mathbf{O}$ be a connected subset of $\mathbf{P}$, and $\mathbf{D}$ be a connected subset of $\mathbf{Q}$, where $\mathbf{D} = \xi(\mathbf{O})$. Let $\partial\mathbf{O}$ be the end points (boundary) of $\mathbf{O}$. We call $\zeta\colon \mathbf{O} \to \mathbf{D}$ a *link* if ζ is continuous, order-preserving, and $\zeta(\mathbf{x}) = \xi(\mathbf{x})$ for each $\mathbf{x} \in \partial\mathbf{O}$. The sets $\mathbf{O}$ and $\mathbf{D}$ are called *origin interval* and *destination interval*, respectively.

A link ζ allows for a level of approximation of ξ on some subsets of the domain of ξ. Let $\mathring{\mathbf{O}}$ be the interior of $\mathbf{O}$. For each point $\mathbf{p} \in \mathring{\mathbf{O}}$, we allow the mapped points $\zeta(\mathbf{p})$ to shift from its original $\xi(\mathbf{p})$, except at the boundaries of $\mathbf{O}$. Although ζ is an approximation, it is required to preserve the ordering of streamlines. The property of *order preservation* handles this notion by disallowing links that have streamlines that cross. To enforce order-preservation, first note that since $\mathbf{O}$ is a connected subset of $\partial\Delta$, it can be parameterized as a single interval. Similarly, since ζ is continuous, only those links are built where $\xi(\mathbf{O})$ is a single interval, again parameterizable. As long as the ordering within this parameterization is preserved, a link is considered valid. Thus, the property of order-preservation leads to the following theorem.

Theorem 6.1 (Consistent Streamlines)**.** Order-preserving links always produce consistent (pairwise disjoint) streamlines.

Chapter 7 will discuss a practical implementation for consistent streamline tracing by exploiting this property of order preservation. The current chapter continues to discuss the definition of links. Since each ζ may be defined only on a subset of $\mathbf{P}$, to completely approximate ξ, only a set of links is needed, leading to the following definition.

Definition 6.9 (Approximate Edge Map)**.** An *approximate Edge Map* ξ^* of ξ is a collection of n links $\zeta_i : \mathbf{O}_i \to \mathbf{D}_i$, such that the collections $\{\mathbf{O}_i\}$ and $\{\mathbf{D}_i\}$ form partitioning of $\mathbf{P}$ and $\mathbf{Q}$, respectively.

The boundary of a triangle can be subdivided into links in many ways to form such a partitioning, thus leading to many possible approximations of ξ. Since the boundaries of each link are "snapped" to require $\zeta(\mathbf{x}) = \xi(\mathbf{x})$, using more links enforces more accuracy. Links can be created by *merging* the origin and destination sets of adjacent OD pairs such that the origin and destination intervals remain connected sets and the orientation of the link is preserved. It turns out that this merging process can be expanded only so much, until specific points where the continuity of the flow breaks. At this maximal amount of merging, a minimal set of links in the approximate Edge Map is obtained.

Definition 6.10 (Base Edge Map)**.** A *base Edge Map* ξ^* of Δ is an approximate Edge Map with minimum number of links. Links of a base Edge Map are called *base links.*

The merging of OD pairs into a single link cannot be done across transition points, sepx points, and images ITPs, because the orientation of a link cannot be preserved across transition points while sepx points and ITP images break the continuity of the map. Thus, the intervals in a base Edge Map are bounded by the transition points, sepx points, and the image points of ITPs. A base Edge Map can be readily constructed by identifying these points and splitting the boundary of a triangle at these points into intervals. The intervals can be paired up into links using connectivity information. Figure 6.4(a) shows the base Edge Map (and the corresponding Time-Edge Map) for a regular triangle highlighting links in different colors.

6.2.4 Inclusion of Critical Points in Edge Maps

So far, this discussion has focused on regular triangles, in which case, a complete partition of of the boundary of the triangle into links is possible. The definition of an Edge Map ξ serves the theoretical purpose of classifying the points on $\partial\Delta$ into OD pairs. However, as discussed above, certain points on $\partial\Delta$ can be classified as unmapped inflow, unmapped outflow, or sepx points; they do not participate in any OD pair. It follows from the definition of the link that these points cannot be included in either origin or destination intervals; hence, they do not get included in any link. Connected subsets of such points are called *unmapped intervals.* The existence of such unmapped points and unmapped intervals is due to the existence of critical points at the interior of the triangle. To facilitate practical applications such as streamline integration, Edge Maps must account for critical points as well.

6.2.4.1 Forward and backward Edge Maps. A *forward Edge Map* $\xi^+ : \mathbf{P} \to \Delta$ is defined such that, given a point $\mathbf{p}$ where a streamline enters the triangle, the map gives the unique point where it exits the triangle. If a critical point $\mathbf{c}$ exists inside the triangle, the flow may never exit, and ξ^+ returns the location of $\mathbf{c}$. Thus, the range of ξ^+ can include the interior of the triangle. On the points on the boundary where flow does not enter the triangle, but instead, exits it, a *backward Edge Map* $\xi^- : \mathbf{Q} \to \Delta$ is defined. For a point $\mathbf{q}$ on the boundary of the triangle, $\xi^-(\mathbf{q})$ describes the unique point where flow entered the triangle on its path to $\mathbf{q}$, or the critical point from which it originated.

It is important to remind the reader that whereas ξ is always a bijection, whose inverse ξ^{-1} represents the Edge Map of the inverted flow, ξ^+ and ξ^- are bijective if and only if

there is no critical point in the triangle. For such triangles, $\xi^- = (\xi^+)^{-1}$, since for points $\mathbf{p}, \mathbf{q} \in \partial\Delta$, $\xi^+(\mathbf{p}) = \mathbf{q}$ if and only if $\xi^-(\mathbf{q}) = \mathbf{p}$. As for triangles with critical point, this inverse relationship does not hold because ξ^+ and ξ^- are no more one-to-one maps. In either case, ξ^+ and ξ^- completely describe the behavior of the flow through the triangle.

6.2.4.2 Forward and backward Time-Edge Maps. If the triangle contains a critical point $\mathbf{c}$, the origin points that flow into the critical point take an infinite amount of time to reach the critical point. Edge Maps replace the infinite time with an approximation at the endpoints $\mathbf{p}$ of the links, $t = \|\mathbf{p} - \mathbf{c}\|/|\vec{v}_\mathbf{p}|$, where $\vec{v}_\mathbf{p}$ is the vector at $\mathbf{p}$. During the analysis or visualization of a streamline, this approximation assumes a constant speed of the particle between $\mathbf{p}$ and $\mathbf{c}$, and is important to ensure a computationally feasible termination of the streamlines going to critical points. The error due to this approximation is limited to the last step of propagation. The forward Time-Edge Map is then extended to $\xi_t \colon \mathbf{P} \to \mathbb{R}$, such that for an origin point, it defines the time taken to reach the destination point, that is, $\xi_t(\mathbf{p}) = t$.

6.3 Equivalence Classes of Edge Maps

As discussed in the previous section, an Edge Map ξ can be approximated in a number of ways using different sets of links. Using the concept of a base Edge Maps, these different Edge Maps can be reduced to the same base Edge Map. Thus, a base Edge Map can be thought of as a representative of an infinite number of Edge Maps, each representing the same flow behavior. This motivates a study of all possible base Edge Maps, which leads to a notion of *equivalence classes* for the Edge Maps that represent all possible types of linearly varying vector fields.

Whereas the notion of reducing any Edge Map to its base Edge Map is well defined, it still needs to be defined how two base Edge Maps can be considered equivalent in order to declare them as belonging to the same class. Before discussing equivalence, the following enumerates the important features of a map required to compare two maps:

- *Pairing* of intervals $(\mathbf{O}_i, \mathbf{D}_i)$ as links.
- *Orientation* of links $(\mathbf{O}_i \to \mathbf{D}_i)$.
- *Cyclic order* of all intervals (including unmapped) on $\partial\Delta$.

To study equivalence, a mixed graph can be constructed for any Edge Map ξ approximated by ξ^* and the set of links $\{\zeta_1, \zeta_2, \ldots, \zeta_n\}$. The goal is to capture the features listed above into a *mixed graph*, $\mathsf{G}(\mathsf{V}, \mathsf{E_U}, \mathsf{E_D})$, as defined below.

- *Nodes*: $\mathsf{V} = \{\mathbf{I}_k\}$ where $\mathbf{I}_k$ is either an origin interval $\mathbf{O}_i$, a destination interval $\mathbf{D}_i$, an unmapped interval $\mathbf{U}_j$, or a sink or source $\mathbf{c}$ in $\mathring{\Delta}$.
- *Undirected Edges*: $\mathsf{E}_\mathsf{U} = \{\{\mathbf{I}_1, \mathbf{I}_2\}\}$ where $\mathbf{I}_1$ and $\mathbf{I}_2$ are adjacent on $\partial\Delta$.
- *Directed Edges*: $\mathsf{E}_\mathsf{D} = \{(\mathbf{I}_1, \mathbf{I}_2)\}$ where either $\zeta_i \colon \mathbf{I}_1 \to \mathbf{I}_2$ (corresponding to a link); $\mathbf{I}_1 = \mathbf{U}_j$ and $\mathbf{I}_2 = \mathbf{c}$ ($\mathbf{I}_1$ is an unmapped inflow interval); or $\mathbf{I}_1 = \mathbf{c}$ and $\mathbf{I}_2 = \mathbf{U}_j$ ($\mathbf{I}_1$ is an unmapped outflow interval).

An example of such a graph is shown in Figure 6.5. Although unmapped intervals do not participate in the Edge Map, they have a structural bearing on the map, and consequently on the mixed graph. They separate links that would otherwise appear contiguous and could be merged. In the case of a saddle, the sepx points are also unmapped. However, they need not have an explicit place in the graph as their existence—and consequently the existence of a saddle in $\mathring{\Delta}$—can be inferred from its neighboring intervals.

Definition 6.11 (Equivalent Edge Maps)**.** Two Edge Maps are considered *equivalent*, if the mixed graphs of their base Edge Maps are isomorphic.

Under this definition of equivalence, a group-theoretic approach for counting all possible equivalence classes of base Edge Maps is presented next. We use the Groups, Algorithms, and Programming (GAP) system [184] for the determination of equivalence. There are three important steps: (1) generation of initial sample space, (2) restricting this space to those possible in linearly varying flow, and (3) identifying equivalences.

6.3.1 Enumeration of All Possible Edge Maps

The analysis for all possible equivalence classes starts with all possibilities for links and unmapped intervals, by disallowing an interval from spanning beyond an edge. This simplifying assumption helps enumerate the initial sample space. Later, this assumption will be removed by removing the splits caused by vertices, thus allowing links to span across

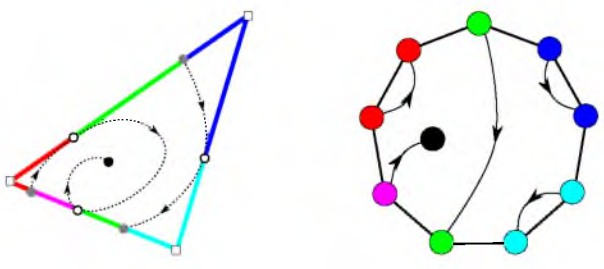


Figure 6.5. Construction of mixed graph from an Edge Map. Points of the magenta interval do not form OD pairs since they flow directly to a critical point.

them. From the definition of base links, and given the property that all links of a base Edge Map are bounded by transition points, sepx points, and ITP images, any base Edge Map has a bounded number of links and unmapped intervals.

Let a, b, c symbolically represent the edges of Δ and d be either the sink or source in $\mathring{\Delta}$, if it exists. For an interval $\mathbf{I}$, its *location*, $\text{loc}(\mathbf{I})$, can be any of the four values in $\mathsf{L} = \{a, b, c, d\}$. The following lemma shows that in G, no two directed edges having their origin nodes in the same location can have their destination nodes in the same location.

Lemma 6.1 (Edge-Level Links)**.** Let $\mathsf{G}(\mathsf{V}, \mathsf{E_U}, \mathsf{E_D})$ be a mixed graph of an Edge Map where no node $\mathbf{I}_i$ spans more than one edge of $\partial\Delta$, and let $\text{loc}(\mathbf{I}_i)$ be the location of $\mathbf{I}_i$. Consider two directed edges $(\mathbf{I}_1, \mathbf{I}_2), (\mathbf{I}_3, \mathbf{I}_4) \in \mathsf{E_D}$. If $\text{loc}(\mathbf{I}_1) = \text{loc}(\mathbf{I}_3)$ and $\text{loc}(\mathbf{I}_2) = \text{loc}(\mathbf{I}_4)$, then $(\mathbf{I}_1, \mathbf{I}_2), (\mathbf{I}_3, \mathbf{I}_4)$ can be merged.

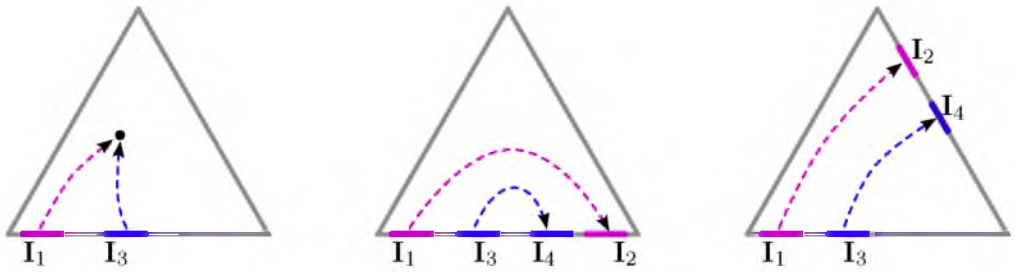
Proof. Let a, b, c be the edges of Δ and d the critical point in $\mathring{\Delta}$. There exist four possibilities of locations for the four intervals, as illustrated in Figure 6.6.

1. $\text{loc}(\mathbf{I}_1) = \text{loc}(\mathbf{I}_3) = \text{loc}(\mathbf{I}_2) = \text{loc}(\mathbf{I}_4) = d$.
   $\implies$ all intervals represent a critical point. This is an impossible case since the one critical point can be either a source or a sink, but not both.
2. $\text{loc}(\mathbf{I}_1) = \text{loc}(\mathbf{I}_3) = d$; and $\text{loc}(\mathbf{I}_2) = \text{loc}(\mathbf{I}_4) \in \{a, b, c\}$.
   $\implies$ $\mathbf{I}_1$ and $\mathbf{I}_3$ represent a critical point.
3. $\text{loc}(\mathbf{I}_1) = \text{loc}(\mathbf{I}_3) \in \{a, b, c\}$; and $\text{loc}(\mathbf{I}_2) = \text{loc}(\mathbf{I}_4) = d$.
   $\implies$ $\mathbf{I}_2$ and $\mathbf{I}_4$ represent a critical point.
4. $\text{loc}(\mathbf{I}_1) = \text{loc}(\mathbf{I}_3) = \text{loc}(\mathbf{I}_2) = \text{loc}(\mathbf{I}_4) \neq d$.
   $\implies$ $\mathbf{I}_1$, $\mathbf{I}_2$, $\mathbf{I}_3$ and $\mathbf{I}_4$ lie on the same edge.
5. $\text{loc}(\mathbf{I}_1) = \text{loc}(\mathbf{I}_3) = e_1 \in \{a, b, c\}$; and $\text{loc}(\mathbf{I}_2) = \text{loc}(\mathbf{I}_4) = e_2 \in \{a, b, c\} - \{e_1\}$.
   $\implies$ $\mathbf{I}_1$ and $\mathbf{I}_3$ lie on one edge, and $\mathbf{I}_2$ and $\mathbf{I}_4$ lie on a different edge.

Since case 3 is the same as case 2 in the inverted flow, only cases 3, 4, and 5 will be considered.

From the discussion in Section 6.2.3, it is known that adjacent OD pairs can be merged except across sepx points, TPs, and ITP images. Therefore, there must exist one of these types of points between $\mathbf{I}_1$ and $\mathbf{I}_3$, and between $\mathbf{I}_2$ and $\mathbf{I}_4$; otherwise they can be merged.

If both $\mathbf{I}_1$ and $\mathbf{I}_3$ lie on $\partial\Delta$ (that is, not both of them represent a critical point), then they cannot be contiguous, that is, $\{\mathbf{I}_1, \mathbf{I}_3\} \notin \mathsf{E_U}$; otherwise they can be merged. Similarly, $\{\mathbf{I}_2, \mathbf{I}_4\} \notin \mathsf{E_U}$, if both $\mathbf{I}_2$ and $\mathbf{I}_4$ lie on $\partial\Delta$. Suppose there exists a subset $\mathbf{I}_5 \in \partial\Delta$ that



(a) Cases 2 and 3 (b) Case 4 (c) Case 5

Figure 6.6. Flow patterns illustrated to prove Lemma 6.1. The four possibilities of locations of the four intervals described in the proof.

lies between $\mathbf{I}_1$ and $\mathbf{I}_3$, and a subset $\mathbf{I}_6 \in \partial\Delta$ that lies between $\mathbf{I}_2$ and $\mathbf{I}_4$. Then $\mathbf{I}_5$ is an inflow interval; otherwise two TPs will exist on either side. This creates more than one TP on the same edge—a scenario disallowed by Property 6.2. Similarly, $\mathbf{I}_6$ must be an outflow interval.

Since streamlines in $\mathbf{I}_5$ and $\mathbf{I}_6$ are bounded by $(\mathbf{I}_1, \mathbf{I}_2)$ and $(\mathbf{I}_3, \mathbf{I}_4)$, irrespective of their locations, there cannot be a TP on $\mathbf{I}_5$ or $\mathbf{I}_6$, and, therefore, ITP images can be ruled out as well. Finally, sepx points are impossible since all four sepx points will have to lie on $\mathbf{I}_5$ and $\mathbf{I}_6$, which would imply a TP on them. Thus, under the given constraints, $(\mathbf{I}_1, \mathbf{I}_2)$ and $(\mathbf{I}_3, \mathbf{I}_4)$ can be merged. □

From Lemma 6.1, it follows that there is a maximum of 13 directed edges in the graph of any Edge Map with edge-level links, as each pair in $\{\mathsf{L} \times \mathsf{L}\} = \{\{a, b, c, d\} \times \{a, b, c, d\}\}$ can appear only once, and the pair (d, d) cannot appear because d is either a source or a sink, but not both. Moreover, no pair $(d, \cdot)$ can coexist with a pair $(\cdot, d)$, again because d is either a source or a sink. Thus, each configuration for any set of directed edges $\mathsf{E_D}$ can be represented as certain elements of the power set of $\{\mathsf{L} \times \mathsf{L}\}$. Consequently, the number of possible configurations for directed edges is 2^{13}.

Under rotations of Δ, certain directed edges are equivalent. For example, $\{(a, b), (b, c)\}$ is equivalent to $\{(b, c), (c, a)\}$. Similarly, $\{(b, b), (c, c)\}$ is equivalent to $\{(c, c), (a, a)\}$ and $\{(a, a), (b, b)\}$. Such equivalences under rotation are eliminated through the action of a permutation group that imparts rotations $(a \to b \to c \to a)$ and $(a \to c \to b \to a)$.

These permutations allow enumeration of the possibilities for the set $\mathsf{E_D}$; however, they still lack any cyclic ordering of nodes to form the undirected edges $\mathsf{E_U}$ in G. Next, all mixed graphs are generated under the constraint that there is a maximum of one TP on the interior of an edge (a consequence of Property 6.1) by grouping all intervals sharing the same edge label. This is done by first generating permutations of all origin/unmapped inflow nodes

and destination/unmapped outflow nodes separately on every edge. All possible orderings of nodes on that edge are then the Cartesian product of these two sets, ultimately constructing the set of all possible mixed graphs (the initial sample space), which numbers more than 100 million.

6.3.2 Restriction to Linearly Varying Flow

The initial enumeration of all graphs is quite large, but contains many invalid graphs. Using the following lemmas based on the linear nature of $\vec{V}$, this initial set can be filtered.

Lemma 6.2 (ITP-vertex)**.** An ITP cannot exist on a vertex of Δ.

Proof. Let vertex $\mathbf{p}_j$ of Δ be an ITP. From the definition of ITP, the curve $\mathbf{x}(\pm t) \in \mathring{\Delta}$ for all $t \in]0, \epsilon]$, such that $\mathbf{x}(0) = \mathbf{p}_j$. The streamline $\mathbf{x}(\pm t)$ is a C^1 continuous curve, whereas the two edges of Δ meeting at $\mathbf{p}_j$ define a corner. By C^1 continuity, $\mathbf{x}(\pm t)$ has a well-defined tangent everywhere, including at $\mathbf{p}_j$, but since the interior angle of Δ at $\mathbf{p}_j$ is less than π, this is not possible for $\mathbf{x}(\pm t)$ being contained in $\mathring{\Delta}$. Thus, the streamline $\mathbf{x}(\pm t)$ must exit Δ to maintain its continuity (as shown in Figure 6.7(a)), contradicting the definition of an ITP, implying an ITP is not possible on a vertex. □

Next, Lemma 6.3 states an important fact that can be used to qualify a critical point based on the existence of an ETP on the interior of an edge of the triangle.

Lemma 6.3 (ETP-Saddle)**.** Let $\mathbf{q} \in \partial\Delta - \{\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k\}$ be an ETP. If there exists a critical point $\mathbf{c} \in \mathring{\Delta}$, then $\mathbf{c}$ is a saddle.

Proof. Let an ETP $\mathbf{x}_e$ exist on an edge a, of Δ. Thus, $\mathbf{x}(\pm t) \notin \Delta$ for $t \in]0, \epsilon]$, where $\mathbf{x}(0) = \mathbf{x}_e$. Let $\mathbf{c} \in \mathring{\Delta}$ be the critical point. From Property 6.1, it is known that there can be only one critical point in $\mathring{\Delta}$. Therefore, if $\mathbf{c}$ is a sink, all streamlines eventually flow to it; if $\mathbf{c}$ is a source, all streamlines emerge from it; and if $\mathbf{c}$ is an orbit, all streamlines are

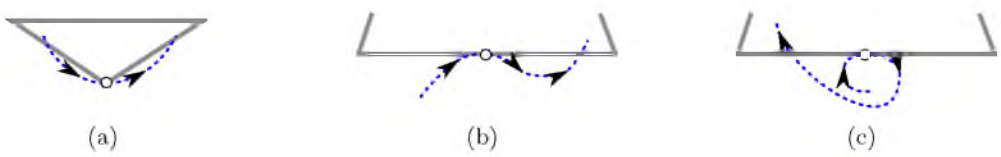


Figure 6.7. Flow patterns illustrated to prove Lemmas 6.2 and 6.3. (a) In Lemma 6.2, a streamline passing through a vertex $\mathbf{p}_j$ of triangle Δ must exit and re-enter Δ. (b) The first case in Lemma 6.3: multiple switches in flow across the edge. (c) The second case in Lemma 6.3: more than one critical point.

closed loops. Thus, if $\mathbf{c}$ is a sink or a source, $\mathbf{x}(t)$ will have to flow to, or emerge from it, which cannot happen without crossing the edge a. There are two ways in which $\mathbf{x}(t)$ can cross a. First, it may make multiple switches of direction (Figure 6.7(b)) over a, violating Property 6.2. Second, one of the images (backward or forward) gets trapped in the enclosure of $\mathbf{x}(t)$ and flows to a critical point inside (Figure 6.7(c)), violating the fact that there is exactly one defined critical point. If $\mathbf{c}$ is an orbit, the closed loop has to enclose $\mathbf{c}$ or it would define another orbit. Thus, again $\mathbf{x}(t)$ is forced to cross a, violating Property 6.2. Thus, $\mathbf{c}$ has to be a saddle. □

Based on these lemmas, the following rules are enforced on all mixed graphs:

1. Directed edges of G cannot intersect (since streamlines can never intersect).
2. ITP cannot exist on vertices (by Lemma 6.2).
3. ETPs cannot exist on edges when there is source or sink (by Lemma 6.3 and Property 6.1).

Using the combinatorial structure of the graph, some undirected edges of a mixed graph can be labeled as either ITPs or ETPs.

Definition 6.12 (Combinatorial Transition Points)**.** For an undirected edge $\{\mathbf{I}_1, \mathbf{I}_2\}$ in G, if $\mathbf{I}_1$ and $\mathbf{I}_2$ switch between inflow to outflow, then $\{\mathbf{I}_1, \mathbf{I}_2\}$ is a *combinatorial TP*. Moreover, if $\mathbf{I}_1$ and $\mathbf{I}_2$ do not have a directed edge between them in $\mathsf{E_D}$, then the TP is a *combinatorial ITP*. Similarly, if $\mathbf{I}_1$ and $\mathbf{I}_2$ have a directed edge between them, then the TP is a *combinatorial ETP*.

Images of an ITP are directly implied by its directed edges. It should be noted that in very few cases, combinatorial ETPs actually correspond to an interior flow that wraps to itself, creating an orbit in the triangle. These cases are not distinguished by the Edge Maps, but can appear only if there is no source or sink within $\mathring{\Delta}$.

Although the analysis described above does not label all the undirected edges, it is sufficient to enforce the aforementioned rules to restrict the sample space of mixed graphs to under 1000. After labeling combinatorial ITPs and ETPs, the second and third rules can be enforced directly. The first rule is enforced using a stack method of detecting intersections between links. In a valid map, any link that has its intervals on $\partial\Delta$ divides the triangle into two halves such that no other link has intervals in both the halves, thus forming a last in, first out sequence on $\partial\Delta$ equivalent to pairing parentheses in a mathematical expression.

6.3.3 Identification of Equivalence Classes

For the final equivalence computation in GAP, contiguous directed edges are merged so that all mixed graphs are reduced to base Edge Maps. Next, the dihedral group for all graph symmetries as well as the permutation group for all labelings of nodes are used. The dihedral group is useful for comparing the mixed graphs under all rigid transformations. The permutation group also includes inversion of the flow direction of directed edges to compare similarities in base Edge Maps under global inversion of flow. Using GAP, all graphs that fall in the same orbit under its actions are enumerated. Each set represents an equivalence class for the graphs. This process produces 43 map classes.

Since the mixed graph definition does not explicitly encode saddles or orbits, the following lemmas are then used to manually invalidate more cases, many of which are challenging to encode within GAP, but are simple to remove by inspection. In particular, the following lemmas study the behavior of ITPs.

Lemma 6.4 (ITP-Saddle)**.** If there is a saddle in $\mathring{\Delta}$, then an ITP cannot exist on $\partial\Delta$.

Proof. Let an ITP $\mathbf{x}_i$ exist on an edge a of Δ. Thus, $\mathbf{x}(\pm t) \in \mathring{\Delta}$ for $t \in]0, \epsilon]$, where $\mathbf{x}(0) = \mathbf{x}_i$. Let $\mathbf{c} \in \mathring{\Delta}$ be the critical point. To prove the lemma by contradiction, assume that $\mathbf{c}$ is a saddle. It follows that all streamlines including $\mathbf{x}(t)$ are parallel to the separatrices of the saddle as $t \to \pm\infty$. However, $\mathbf{x}(t)$ must cross an edge of Δ multiple times to stay parallel to the separatrices in those limits of t, contradicting Property 6.2. This is true since at least one separatrix intersects every edge of Δ forcing multiple intersections of $\mathbf{x}(t)$ with the edge to which $\mathbf{x}_i$ belongs. Hence, $\mathbf{c}$ cannot be a saddle. The cases for saddle sectors are shown in Figures 6.8(a) and 6.8(b). □

Additionally, in the presence of an ITP, Lemma 6.5 guarantees that a critical must be enclosed by the streamline extending from the ITP.

Lemma 6.5 (ITP-CP Enclosure)**.** Let $\mathbf{x} \in \partial\Delta$ be an ITP with images $\mathbf{x}_b, \mathbf{x}_f \in \partial\Delta$, and $\mathbf{c}$ be a critical point in $\mathring{\Delta}$. Then, $\mathbf{c}$ must lie within the area bounded by the streamline $\mathbf{x}(t)$ and $\partial\Delta$ where $t \in [a, b]$ such that $\mathbf{x}(a) = \mathbf{x}_b$ and $\mathbf{x}(b) = \mathbf{x}_f$.

Proof. Let an ITP $\mathbf{x}_i$ exist on an edge a of Δ and $\mathbf{c}$ be a critical point in $\mathring{\Delta}$. To prove the lemma by contradiction, assume that $\mathbf{c}$ exists outside the enclosure of $\mathbf{x}(t)$ in $\mathring{\Delta}$, where $\mathbf{x}(0) = \mathbf{x}_i$. It will be shown that this leads to a contradiction. Since $\mathbf{x}_i$ is an ITP, $\mathbf{c}$ cannot be a saddle (by Lemma 6.4). Let a, b, c be the three edges of Δ. Now, all possible combinations of edges on which $\mathbf{x}_i$, $\mathbf{x}_f$, and $\mathbf{x}_b$ can lie are enumerated.

1. $\text{edge}(\mathbf{x}_i) = \text{edge}(\mathbf{x}_b) = \text{edge}(\mathbf{x}_f)$.

   The only way to realize this case is an orbit such that $\mathbf{x}_i = \mathbf{x}_f = \mathbf{x}_b$. Such an orbit immediately encloses the critical point.

2. $\text{edge}(\mathbf{x}_i) = \text{edge}(\mathbf{x}_b)$ or $\text{edge}(\mathbf{x}_i) = \text{edge}(\mathbf{x}_f)$.

   These two cases are symmetrical under inversion of flow. If either $\mathbf{x}_f$ or $\mathbf{x}_b$ lie on the same edge as $\mathbf{x}_i$, by Property 6.2, the other image gets enclosed into $\mathbf{x}(t)$, and has to flow to a critical point.

3. $\text{edge}(\mathbf{x}_i) \neq \text{edge}(\mathbf{x}_b) = \text{edge}(\mathbf{x}_f)$.

   In this case, both $\mathbf{x}_b$ and $\mathbf{x}_f$ lie on the same edge, indicating a switch from inflow to outflow, implying that there exists at least one TP between them including the two points. If one of $\mathbf{x}_f$ and $\mathbf{x}_b$ is a TP, it has to be an ITP since part of its streamline is already in $\dot{\Delta}$. Due to Property 6.2, the other image of this ITP has to flow into a critical point in the enclosure of $\mathbf{x}(t)$. If a point in the interior is a TP, this TP cannot be an ETP since it has already been established that $\mathbf{c}$ is not a saddle and an ETP would violate Lemma 6.3. Now one of the images of this ITP can flow back to the same edge, but the other has to flow to $\mathbf{c}$. Thus, $\mathbf{c}$ is in the enclosure of $\mathbf{x}(t)$, the streamline of the ITP.

4. $\text{edge}(\mathbf{x}_i) \neq \text{edge}(\mathbf{x}_b) \neq \text{edge}(\mathbf{x}_f)$.

   In this case, it is always possible to construct a new triangle by connecting $\mathbf{x}_f$ and $\mathbf{x}_b$ as an edge and arbitrarily choosing a third point such that $\mathbf{c}$ still lies in the interior of the new triangle as shown in Figure 6.8(c). An argument identical to case 3 can now be applied.

Thus, in all four cases, $\mathbf{c}$ must lie in the enclosure of the streamline of the ITP. □

As a result of these lemmas, the following rules are enforced on the remaining cases to

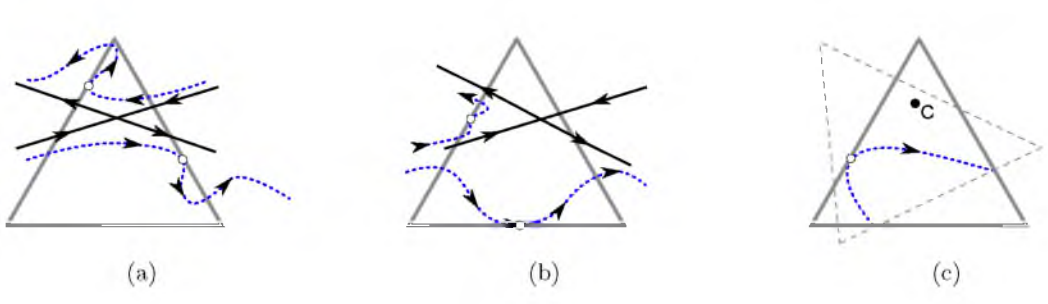


Figure 6.8. Flow patterns illustrated to prove Lemmas 6.4 and 6.5. (a) and (b) Lemma 6.4: Cases of saddle sectors intersecting $\partial\Delta$ invalidating the existence of an ITP on $\partial\Delta$. (c) Construction of a new triangle as in Lemma 6.5 to place both the forward and backward images of an ITP on the same edge.

get the final equivalence classes:

- The graphs should not violate Lemma 6.4.
- The graphs should not violate Lemma 6.5.
- There can be a maximum of one critical point. If there is a sink or a source in the map, there cannot be a sepx point (Property 6.1, since a sepx point means there must additionally be a saddle).
- There have to be exactly four sepx points or none at all (by defn. of a linear saddle).

To enforce the above rules, *combinatorial sepx* points must be identified in the graphs. Since all contiguous directed edges have been merged, the graphs represent the base Edge Maps. Thus, any unlabeled undirected edges have to correspond to sepx points, since all ETP, ITP, and ITP images have already been labeled. Knowing the labeling of each undirected edge allows invalidating 20 of the 43 cases produced by GAP. 23 classes persist after the invalidations, visualized in Figure 6.9. Corresponding PL vector fields are also shown to realize each of these cases, confirming the following result.

Theorem 6.2. Under Definition 6.11 for equivalence, there are 23 equivalence classes of Edge Maps for triangles in a linearly varying vector field.

6.4 Computation of Edge Maps

An Edge Map (forward and backward) can be encoded concisely as a collection of links of a triangle, such that the intervals are nonintersecting other than at their endpoints, and cover the entire boundary of the triangle (as illustrated in Figure 6.4). If there is a critical point present in the triangle, some links may include the critical point as a source or destination interval. Thus, to store the Edge Map for a triangle, only a collection of links needs to be stored.

As discussed in Section 6.2, intervals are constructed by merging adjacent origin points whose destinations are also adjacent. At the maximum level of merging (base Edge Maps), the intervals are bounded by the following three types of points:

- vertices of the triangle and their images (Figure 6.10(a)),
- transition points, and the images of internal transition points (Figure 6.10(b)), and
- *sepx points*, where the separatrices of a saddle exit or enter (Figure 6.10(c)).

To generate Edge Maps, these points need to be identified, requiring critical point identification (discussed in Section 7.2). The advection of vertices, ITPs, and saddle points is

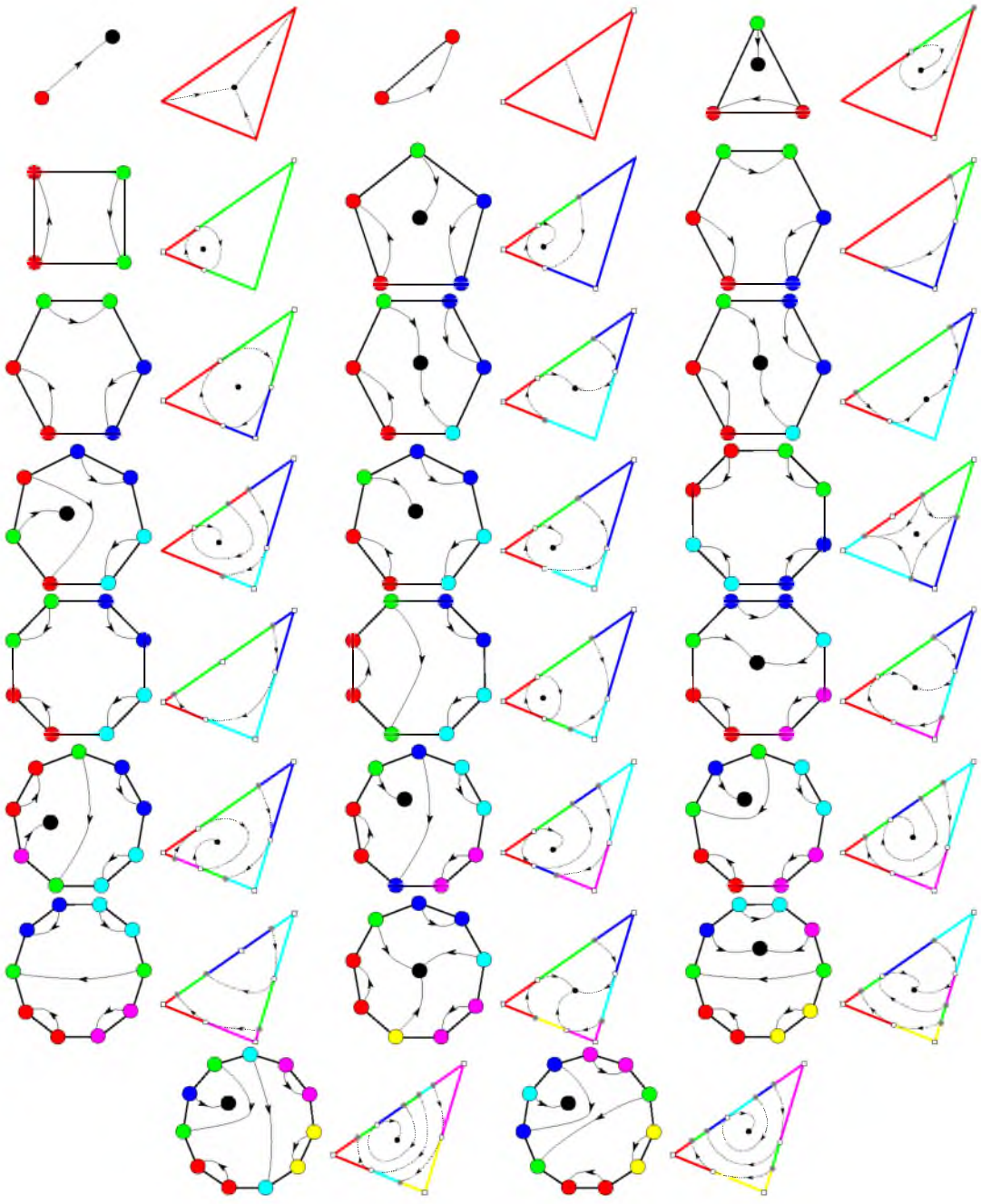


Figure 6.9. The 23 equivalent classes of mixed graphs, along with one possible rendition of Edge Map for PL flow for each class, in the order of increasing number of links.

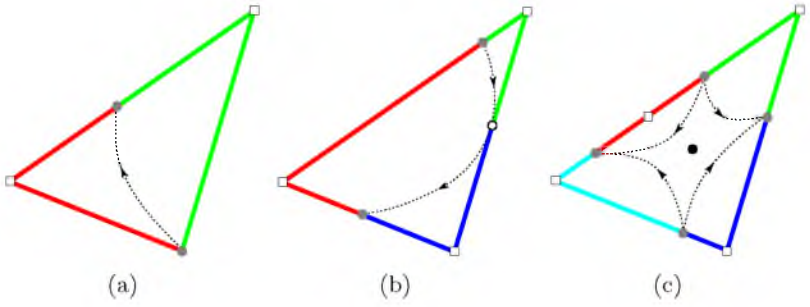


Figure 6.10. The types of points (and their images) required to split the boundary of a triangle to create Edge Maps. (a) A forward vertex image (grey dot) of the lower right vertex; (b) An internal transition point (white circle) and its forward and backward image (grey dots); (c) The sepx points (grey dots) of a saddle point (black dot), and a transition point from external flow (white square). Note that in most cases, vertices also act as external transition points.

performed using the local exact method (LEM) [108]. The LEM is an analytical solution that computes the streamlines accurately without any floating-point truncation errors. During the generation of Edge Maps, the LEM also computes the time taken by these points to reach their images. This time information can be stored in Edge Maps for every point that is advected. Thus, the endpoints of origin intervals are assigned the time they take to reach the endpoints of their respective destination intervals.

Algorithm 6.1 describes the construction of the base Edge Map for a triangle without a critical point. When the triangle has a critical point, the algorithm is similar except that there can be additional cuts (from separatrices) and the critical point itself can act as an interval.

Algorithm 6.1 ConstructEdgeMap($\vec{V}, \Delta$).

---

1. Identify the transition points on $\partial\Delta$. Advect the internal transition points forward and backward to find their images. *(There can be at most six transition points in a triangle: one per edge and one per vertex.)*

2. Advect any vertices of Δ that are not transition points forward (respectively backward) to find their images. *(The transitions points, vertices, and their images cut $\partial\Delta$ into intervals of unidirectional flow.)*

3. Using the direction (inflow/outflow) and connectivity implied by advecting, pair intervals to form links. *(Collection of these links compose the Edge Map.)*

6.5 Streamlines Computation Using Edge Maps

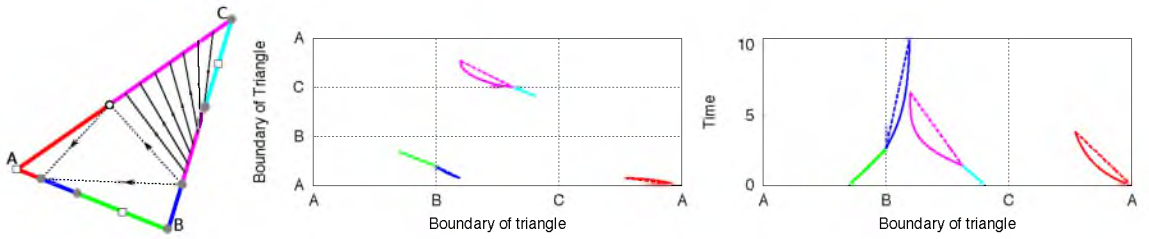
As discussed above, Edge Maps can be created by advecting the vertices, ITPs, and sepx points of Δ dividing $\partial\Delta$ into links. For a compact and practical representation, only the end points of the links are stored to represent the entire flow through Δ. Thus, although the end points of the links represent the flow accurately, an approximation must be made to represent the flow at the interior of the links. In this work, we consider a linear approximation of Edge Maps within the links, which satisfies the property of order preservation up to floating-point errors. The encoding of flow as Edge Maps ultimately allows us to determine structural properties of the flow through the triangle using a simple lookup, consequently, allowing the computation of flow-based properties efficiently.

In particular, streamlines can be approximated on a per-link level by interpolating between origin and destination intervals. Hence, Edge Maps provide a way to approximate streamlines. As Figure 6.11 shows, for an origin point $\mathbf{p}$ on $\partial\Delta$, its path to its destination $\hat{\mathbf{q}}$ can be approximated by linearly interpolating in the origin interval corresponding to $\mathbf{p}$ and projecting it to the same barycentric coordinate in the destination interval.

Using the precomputed Edge Maps, any numerical integration to calculate streamlines given by Equation (6.1) can be replaced by a simple lookup:

$$\mathbf{x}_{n+1} = \xi(\mathbf{x}_n); \qquad \text{and} \qquad t_{n+1} = t_n + \xi_t(\mathbf{x}_n) \tag{6.2}$$

Thus, at each lookup, an important property is preserved, that origin intervals are mapped onto the same destination intervals they would have in the original PL flow in a consistent manner. The linear approximation within the links discards the flow behavior of the interior of triangles, but preserves enough information to maintain consistency of



(a) (b) (c)

Figure 6.11. Linear approximation of the Edge Map from Figure 6.4 for the triangle from Figure 6.2. The plots of ξ and ξ_t are reproduced (solid) from Figures 6.4(b) and 6.4(c), and compared with their linear approximation (dashed). The deviation between the two curves is the error of Edge Maps due to this approximation, shown in Figure 6.12.

streamlines. Furthermore, the streamlines computed using Edge Maps have a bounded error that can be explicitly computed and reduced, as will be discussed next.

6.6 Representation and Reduction of Error

Streamlines are typically traced through numerical integration. From a given starting point, these techniques repeatedly take small steps to approximate the next position in the path. The resulting error is controlled only indirectly by choosing a step-size [79]. Since typically the true streamline is not known, this error cannot be quantified explicitly. Although some schemes are more accurate than others and sophisticated techniques exist to locally adapt the step-size, the indirect control over an unknown error represents a fundamental restriction. On the contrary, Edge Maps represent and control the error in representation explicitly and do not require setting a step-size.

Furthermore, integrating streamlines numerically can also lead to inconsistencies, such as intersecting streamlines and significant differences between forward and backward traced lines. Edge Maps replace integration with a 1D linear mapping that guarantees nonintersecting streamlines and consistency between forward and backward traces up to the floating-point precision of the linear interpolation. This section studies the approximation errors in Edge Maps, and provides tools to quantify and visualize them.

6.6.1 Approximation Errors in Edge Maps

Section 6.5 discussed streamline propagation with a bounded error using Edge Maps as given by Equation (6.2). Since an Edge Map approximates the true exit point $\mathbf{q}$ and the true exit time t of a point $\mathbf{p}$ by linearly interpolating within the link as $\mathbf{q}'$ and t', respectively, it incurs some error. The spatial error can be calculated as the deviation of the exit point given by the map from that given by the exact method ($\varepsilon = \|\mathbf{q} - \mathbf{q}'\|$). Similarly, the temporal error can be calculated as the deviation of the exit time given by the map, from that given by the exact method ($\delta = t - t'$). Figures 6.11(b) and 6.11(c) show this deviation using a comparison between the true flow and its linear approximation using Edge Maps, and Figure 6.12 shows the error plots for the spatial and temporal error. A derivation of the mapping error in Edge Maps is provided in Appendix A.2.1.

The errors for both the spatial and temporal deviations are computed by sampling the link. Figure 6.12 plots the spatial and temporal errors as a function of the boundary of the triangle. The maximum spatial error in a link l is used as the *spatial error of the link*, ε_l. Similarly, the *temporal error of the link* δ_l is defined as the maximum temporal error within

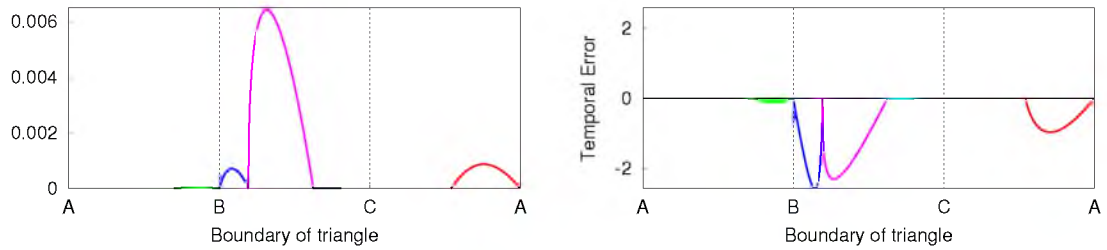


Figure 6.12. The error plots for spatial (left) and temporal (right) errors for the linearly approximated map shown in Figure 6.11. The average edge length of the triangle in consideration is 0.03, and the time taken to travel across the triangle can be inferred from Figure 6.11(c).

a link. To account for the amount of error incurred during the propagation, $\xi_\varepsilon^+ : \mathbf{P} \to \mathbb{R}$ and $\xi_\delta^+ : \mathbf{P} \to \mathbb{R}$ are defined such that

$$\xi_\varepsilon^+(\mathbf{p}) = \varepsilon_l; \qquad \text{and} \qquad \xi_\delta^+(\mathbf{p}) = \delta_l$$

where ε_l and δ_l are the spatial and temporal errors of the link l containing $\mathbf{p}$. Thus, two floating-point values are stored per link to represent the mapping error in Edge Maps.

As explained in Section 6.4, the vertices, saddle separatrices, and transition points are advected to split $\partial\Delta$ into intervals. Since the LEM is used for this advection, the endpoints of the intervals are accurate up to the floating-point precision of the system. Hence, the mapping errors (both spatial and temporal) are zero at the endpoints of links.

In our experiments, it was found that the spatial error is typically unimodal in a link. However, the error can also be bimodal, as shown in Figure 6.13(a). Similarly, the temporal error can be both positive and negative within the same link, as shown in Figure 6.13(b). It must be noted that certain types of flows are less prone to error than others. For example, consider concentric circular orbits or a linear flow where any two streamlines do not diverge from or converge to each other. The mapping error is zero for such a case since the actual flow agrees with the linear mapping within each link. Figure 6.14 corroborates this intuition by testing the Edge Map propagation in a purely rotational flow. Hence, in the absence of mapping error, the propagation using Edge Maps is as accurate as the underlying method for advection used for map generation.

6.6.2 Refinement of Edge Maps

An upper limit to the mapping errors can be imposed by user parameters, $\varepsilon_{\max}$ and $\delta_{\max}$. If for a link, $\varepsilon_l > \varepsilon_{\max}$ or $\delta_l > \delta_{\max}$, the link is split at the point of maximum error to improve the accuracy of the map through the *refinement* of Edge Maps. Figure 6.15 shows

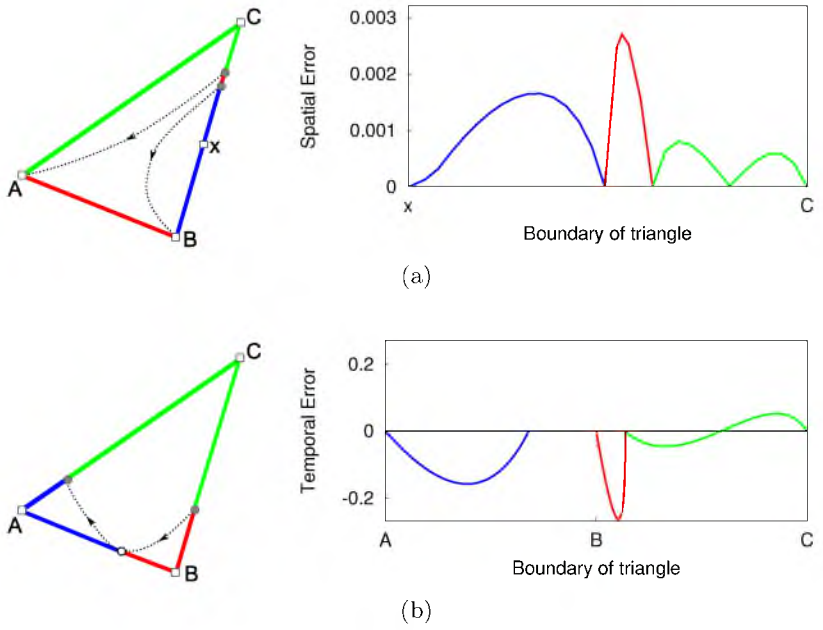


Figure 6.13. Two types of possible behavior of spatial and temporal errors. (a) A (forward) Edge Map with bimodal spatial error in the green link. Note the horizontal axis has been scaled to the range $[x, C]$ to illustrate the error. (b) A (forward) Edge Map with both positive and negative temporal error within the green link. The horizontal axis has been scaled to range $[A, C]$ to illustrate the mapping error. In the destination interval on edge AB, there is no temporal error defined under the forward Edge Map. The average edge length of both triangles in consideration is 0.3.

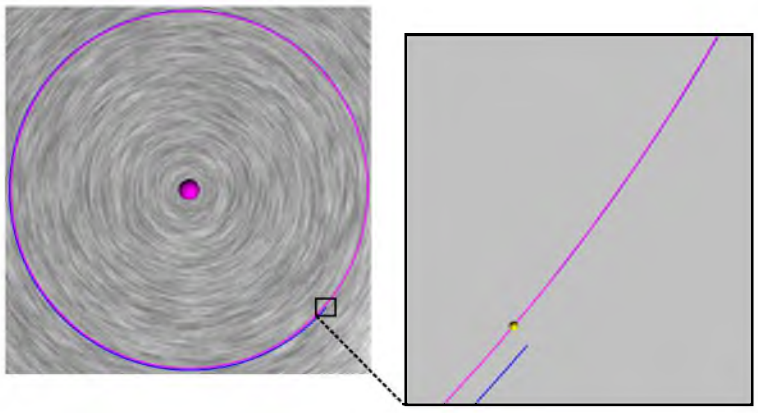


Figure 6.14. Comparison between streamline propagation using Runge-Kutta 4-5 (RK45) (blue) and Edge Maps (magenta) on a vector field defined by a counterclockwise orbit seeded at the same point (yellow). The magenta and blue lines overlap in the beginning, but a substantial deviation in RK45 streamline is observed after only one revolution around the critical point (purple). In the absence of mapping error, the mapped lines are accurate up to floating-point precision.

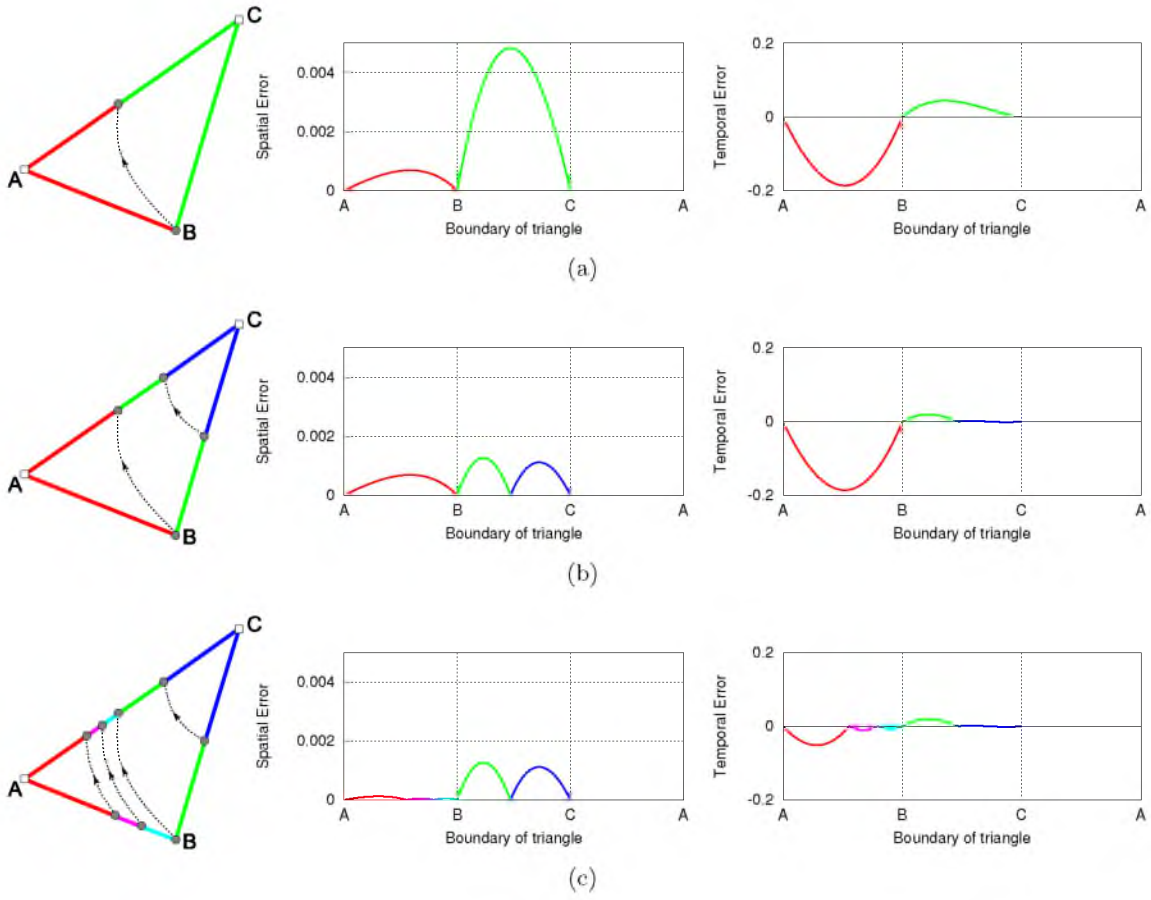


Figure 6.15. Reduction of mapping error through refinement of maps. The length of the edge AC is 0.0354, and the average time taken by a particle to travel across the triangle is 1.7. (a) Base maps and the corresponding errors. (b) Spatial refinement with $\varepsilon_{\max} = 0.003$ splits the green link into two, creating two new links (green and blue) with smaller errors. (c) Temporal refinement with $\delta_{\max} = 0.06$ splits the red link twice, creating three new links (red, cyan, and magenta) with even smaller errors.

the effect of spatial (Figure 6.15(b)) and temporal (Figure 6.15(c)) refinement on the Edge Map shown in Figure 6.15(a). The level of refinement needed for a given flow is subject to the nature and magnitude of flow, and the triangle size. Thus, the computation time and memory consumption of refined maps depend upon these factors.

6.7 Uncertainty Visualization of Spatial Error

Explicit representation of error in Edge Maps enables error visualizations of flow. This section discusses how to generate visualizations of spatial error using Edge Maps.

6.7.1 Expansion of Exit Points

So far, the forward (ξ^+) and backward (ξ^-) Edge Maps have been used as tools to look up the streamlines of individual particles. However, we can also represent the spatial error explicitly by redefining the Edge Maps as a one-to-many map.

$$\xi^+(\mathbf{p}, \omega) = \mathbf{Q},$$

where for an entry point $\mathbf{p}$, instead of a single exit point $\mathbf{q}$ the map gives a range of possible exit points, a segment $\mathbf{Q}$, under the *expansion factor*, ω, as illustrated in Figure 6.16. The length of the segment $\mathbf{Q}$ is directly proportional to the expansion factor. Thus, we call $\mathbf{Q}$ the *expansion* of the exit point $\mathbf{q}$.

The spatial error ε for $\mathbf{p}$ encodes the deviation of its exit point $\mathbf{q}$ defined by the Edge Map from the true exit point $\tilde{\mathbf{q}}$. Therefore, the expansion of the exit point $\mathbf{Q}$ calculated using $\omega = \xi_\varepsilon^+(\mathbf{p})$ provides an upper bound on the possible exit points of $\mathbf{p}$. Furthermore, since the streamlines at the endpoints of the links are accurate, the expansion cannot span across links, and is truncated at the endpoints of the link containing both $\mathbf{p}$ and $\mathbf{q}$.

6.7.2 Streamwaves

The one-to-many mapping given by Edge Maps under the consideration of spatial error can be visualized as a *streamwave*, which is defined as the set of possible destinations that a massless particle may reach when accounting for possible expansions. Alternatively, a streamwave can be seen as the expansion of a streamline due to spatial uncertainties. In the current work, we quantify and visualize the spatial error as streamwaves propagate, by setting $\omega = \varepsilon_l$. However, any other kind of error can be modeled as the expansion ω for streamwaves.

Using Edge Maps, streamwaves can be computed as follows.

$$\mathbf{X}_{n+1} = \xi^+(\mathbf{X}_n, \omega),$$

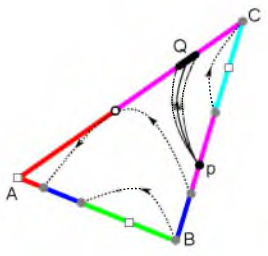


Figure 6.16. Expansion of exit points represents error as a range of possible destinations.

where $\mathbf{X}_0 = \{\mathbf{x}_0\}$ represents the seed point of the wave and $\mathbf{X}_n$ the set of points currently at the front of the wave. Since a streamwave models the spatial error only, the speed of the wave is reflective of the linear time approximation of the Edge Maps. The temporal error in the Edge Maps is ignored. Using traditional techniques to compute streamwaves as a collection of streamlines can become computationally expensive and requires delicate processing in regions of high divergence. Using Edge Maps, however, propagating a wave is only as expensive as the number of links in the triangles currently at the front.

Furthermore, if there exist no bifurcations in a triangle, then only extremes of the range of exit points $\mathbf{X}_{n+1}$ are of interest, and all intermediate points are handled implicitly. For triangles with bifurcation, a streamwave may split into two streamwaves, each of which can be propagated independently.

Figure 6.17 shows streamwaves computed on a 2D flow resulting from the simulation of a homogeneous charge compression ignition (HCCI) engine combustion [88] (see Data B.1.1). This 2D flow defined on a $[640 \times 640]$ domain is incompressible, and exhibits turbulent vortical structures of clockwise and counterclockwise rotations. The computation of Edge Maps for 816,642 triangles in the dataset took 223 seconds and 200 MB. As shown in the figure, a streamwave is the superset of a single streamline, so analyzing only the streamline in the presence of error is an incomplete analysis. Notice how even small amounts of errors ($\approx$ 3.2% and $\approx$ 6.4%) can cause large compounded uncertainties in streamlines. Since

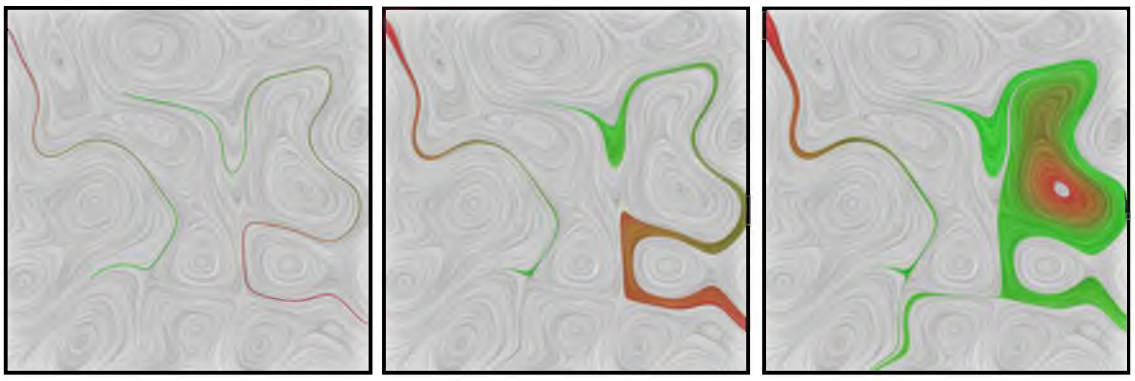


Figure 6.17. Streamwaves in the flow in an HCCI engine [88]. The $[640 \times 640]$ data is mapped to a $[-1, 1] \times [-1, 1]$ plane. Left: Two streamlines – one near a saddle's separatrix. Middle and right: Streamwaves at different levels of error: 0.0001 ($\approx$ 3.2% of horizontal and vertical spatial resolutions) and 0.0002 ($\approx$ 6.4% of horizontal and vertical spatial resolutions). Streamwaves are colored from green to red, showing the distance the flow has propagated as a measure of the number of maps the streamwave has travelled through. Note that the error levels have been exaggerated to illustrate expansion and bifurcation of streamwaves.

expansion of a streamwave in the presence of error may cause it to revisit a certain region, it may need to be truncated so as to avoid going into infinite flow loops. This truncation is consistent with the definition of streamwave: that it visualizes the region that can be visited by a particle (at least once). The shape of streamwave reflects the nature of the underlying flow. A 'linear' streamwave will be obtained if the flow is mostly linear (little or no rotation), for example, Figure 6.18. If the flow is highly rotational, the streamwave revisits certain regions and becomes 'blob-like', for example, in Figure 6.17. The color of the streamwave progresses from green to red as it propagates forward in time, as an indication of the speed of the streamwave.

Streamwaves also present a method to visualize error bounds of other integration techniques. For example, Figure 6.18 shows the integration of a streamline connecting a source to a sink using three different techniques. By showing a streamwave, whose expansion is set larger than the maximum error for Euler integration, a comparison between Euler integration, fourth-order Runge-Kutta, and LEM can be visualized.

6.7.3 Visualization of Fuzzy Topology

Topological structures in vector fields, such as their topological skeleton [89], are one of the key features used to analyze vector field data. Traditionally, the skeleton is computed by tracing four separatrices out of each saddle (two forward and two backward) by com-

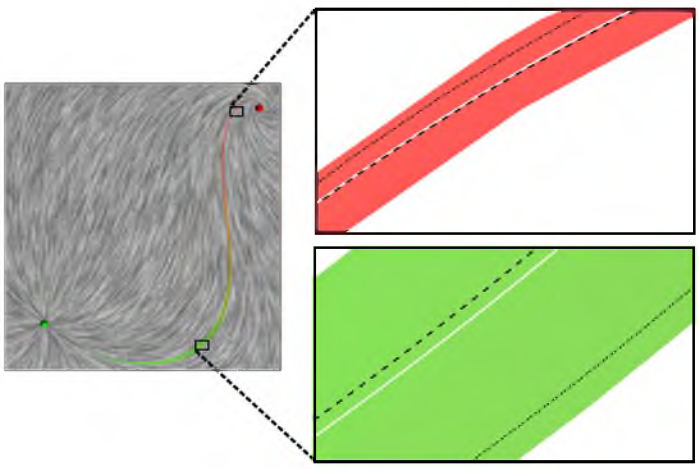


Figure 6.18. Comparison of various integration schemes with streamwaves. A streamline using Runge-Kutta 4 (RK4) (black dashed) using step-size $\hbar = 0.005$, Euler (black dotted) using step-size $\hbar = 0.005$ and local exact method (LEM) (white solid), and a streamwave using Edge Maps with $\omega = 0.0001$ were seeded at the same point. Considering the local exact method to be the ground truth, some deviation is observed in Euler and RK4 streamlines. It is also observed that the streamwave, centered around the LEM streamline, bounds the two erroneous streamlines at all the times.

puting streamlines starting in the directions of the eigenvectors of all the saddles. These separatrices terminate when they arrive at another critical point or leave the boundary of the domain. However, this approach faces challenges since compound integration error can cause the trace to end at an incorrect critical point. In particular, unstable topologies, such as when a pair of saddles is connected by a separatrix, suffer from this form of inconsistency.

Instead, streamwaves can be used to study the robustness of topological representations. By growing a streamwave with $\omega = 0$, in the forward direction from all sources and in the reverse direction from all sinks, a partial topological decomposition of a vector field can be performed that is analogous to stable and unstable manifolds in scalar field topology [48]. These streamwaves are initiated from the segments of the boundary flowing out of and into the triangles containing the sources and sinks, respectively. Although the centers cannot yet be accounted for, streamwaves can provide important information about the structure of the flow. In particular, in the absence of closed orbits, the union of the forward and backward streamwaves creates a covering of the domain similar to the segmentation induced by traditional vector field topology.

However, in a similar construction for $\omega > 0$, each point in the domain may be part of several streamwaves creating a notion of *fuzzy* topology, as shown in Figure 6.19. Setting $\omega = \varepsilon_l$ highlights the regions of the domain that are unstable/uncertain under the approximation errors of Edge Maps as thick red bands, thus providing important information about any potential instabilities in the topological segmentation. In particular, it provides users with an intuitive measure of how certain a given structure is.

To illustrate the new concept of fuzzy topology, streamwaves are compared with the

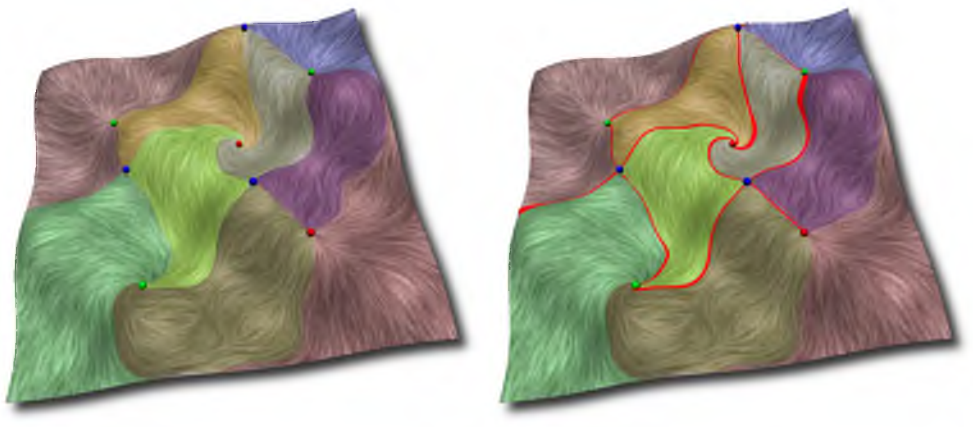


Figure 6.19. Fuzzy topology (right) in a synthetic flow containing three sources (green), two sinks (red), and three saddles (blue). Compare with "certain" toplogy (left). Streamwaves with $\omega = 0$ (left) and $\omega > 0$ (right) are used for such a construction.

traditional scalar field techniques, as visualized in Figure 6.20. Laney et al. [115] performed a topological analysis on the interface surface of a heavy fluid placed above a light fluid to understand the Rayleigh-Taylor instability (see Data B.2.1). In particular, the unstable manifolds of the height function segment the surfaces into "bubbles"—the primary feature of interest. To generate a similar result, the gradient flow of the height function can be used to construct the manifolds using streamwaves. Both techniques provide a similar view of the data, but streamwave-based representation is richer by also showing the inevitable inconsistencies at the boundaries of the bubbles.

Figures 6.21 and 6.22 show, respectively, fuzzy topology computed on surface flows for a diesel engine and a combustion chamber [117] (see Data B.2.2) with refined maps. Since the spatial error in the maps is very low, the fuzzy regions are negligible.

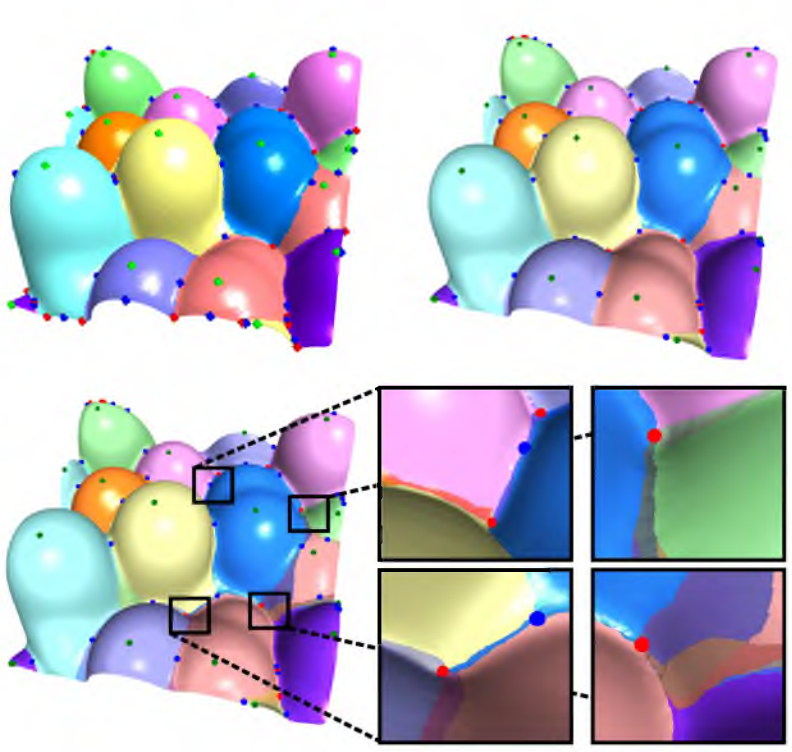


Figure 6.20. Fuzzy topology in Rayleigh-Taylor instability (see Data B.2.1). The bubbles extracted using scalar field topology similar to the work of Laney et al. [115] (top left) are shown side-by-side with the unstable manifolds (streamwaves computed using $\omega = 0$), (right). Bottom row: when the error factor, $\omega > 0$, is accounted for, emerging overlaps are observed, as highlighted in zoomed-in views.

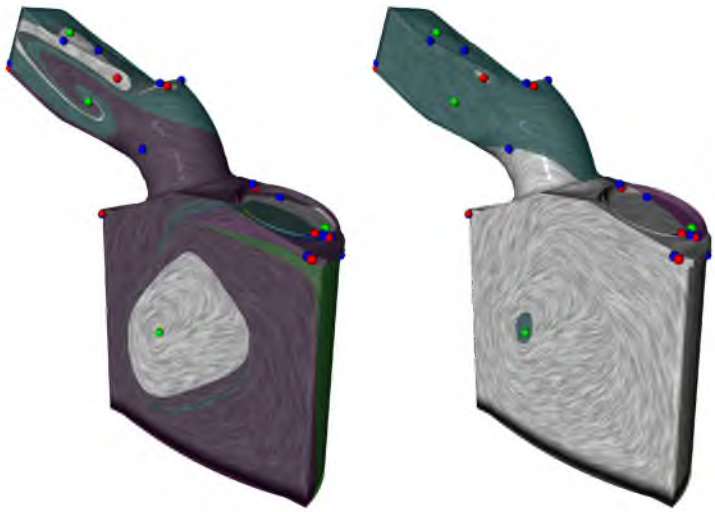


Figure 6.21. Stable (left) and unstable (right) manifolds of the flow defined on the surface of a combustion chamber [117] (see Data B.2.1). The flow has 13 sources (green), 15 sinks (red), and 27 saddles (blue). With sufficiently refined maps, the fuzzy regions are negligible, indicating a more accurate representation.

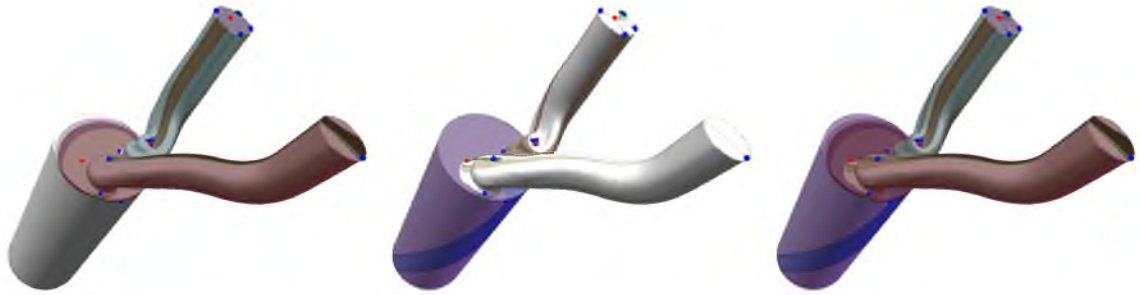


Figure 6.22. Stable and unstable manifolds of the flow defined on the surface of a diesel engine (see Data B.2.1). Stable (left), unstable (middle), and both (right) manifolds.

6.8 Uncertainty Visualization of Temporal Error

For a streamline that is computed assuming error-free propagation, there is a unique time instant given for every position and vice versa. Although all numerical integration techniques have some associated error, however small or big, this error is generally ignored [156]. Using Edge Maps, it is possible to bound and visualize the temporal error in streamlines. The discussion in this section considers only the temporal errors in the map, and assumes no spatial error.

6.8.1 Temporal Error in Streamlines

The temporal error of a link (δ_l) represents the maximum possible deviation in an Edge Map's approximation of time from the true time. The points on the streamline corresponding to a temporal error will span across triangles. Since an Edge Map is not aware of the vector field in neighboring triangles, it cannot return a temporal expansion of points as in the case of the spatial expansion (see Section 6.7.1). Instead, the temporal expansion of points is computed indirectly using the temporal error. When a streamline is integrated using Edge Maps, the temporal error of the links it passes through is accumulated.

$$\delta_{n+1} = \delta_n + \xi_\delta^+(\mathbf{x}_n).$$

Thus, every point $\mathbf{x}_n$ on the streamline has a value of time t_n and a temporal error δ_n.

Figure 6.23 illustrates the accumulation of temporal error along a streamline. Given a point $\mathbf{p}$ on the streamline, the time and the temporal error can be easily found by interpolating the data stored in the streamline. The accumulated temporal error is visualized as an error field along the streamline in Figure 6.24.

6.8.2 Spatial Visualization of Temporal Error

The temporal error δ_n for a point $\mathbf{x}_n$ on a streamline defines a range of possible values of time $[t_n - \delta_n, t_n + \delta_n]$ that can be associated with $\mathbf{x}_n$. This range is shown as the vertical thickness of the error envelope in Figure 6.23. Another way to visualize the temporal error is to understand its spatial manifestation, which gives a more intuitive understanding of flow behavior under error.

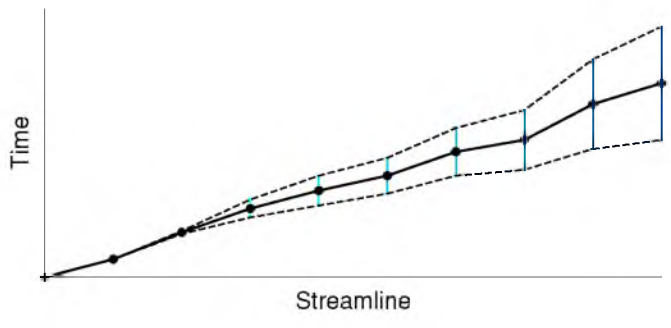


Figure 6.23. Accumulation of temporal error shown as an error plot for a streamline in time space. Every point on the streamline has an associated temporal error, shown by the vertical thickness of the dashed envelope. The vertical thickness of the envelope is twice the amount of temporal error.

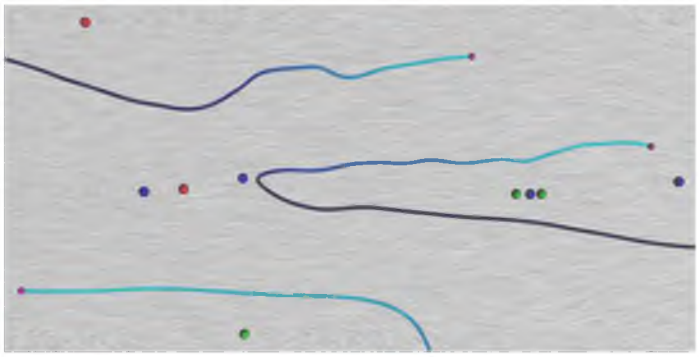


Figure 6.24. Accumulated temporal error in streamlines for the 2D ocean winds flow (see Data B.1.2). The flow contains three repelling foci (green balls), two attracting foci (red ball), and four saddles (blue balls). The seed points of the streamlines are shown as small magenta balls. The color map cyan-dark blue is mapped to increasing temporal error along the streamline. A sharp increase in accumulated errors near the saddle at the center of the field indicates a region of high temporal error.

A streamline generated using Edge Maps can be queried for all the points that could be visited at a given time τ. This query returns every point whose time-range contains τ. However, since an Edge Map streamline contains only the points lying on the edges of the triangulation, the time-ranges between adjacent points on streamline are interpolated to include the points on the interior of triangles. This computation is analogous to intersecting the time envelope (in Figure 6.23), with a horizontal line representing some value of time, and projecting this intersection onto the horizontal axis as shown in Figure 6.25. Thus, a spatial range is obtained that is centered around the true position—the position in absence of temporal error—of the particle at every time instant.

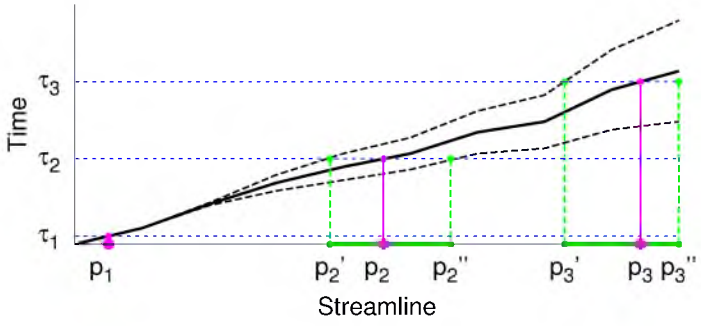


Figure 6.25. Computation of the spatial range corresponding to the temporal error in a streamline. For every time instant (τ_i) represented by a dashed blue horizontal line, its intersections with the temporal error envelope (black dashed) are computed, and then projected onto the horizontal axis (the streamline), giving the spatial range of points ($\mathbf{p}_i'$, $\mathbf{p}_i''$) on the streamline, shown in green. The true position (considering no error) is the projection of the intersection with the actual time (black solid) ($\mathbf{p}_i$), shown in magenta.

Figure 6.26 shows the visualization of the spatial range due to temporal error. The spatial range presents a way to visualize the bounds on the uncertainty in the position of the particle at a given time. Note that the extracted spatial range is only a spatial manifestation of temporal error. An equal amount of temporal error will produce a longer spatial range for higher vector magnitudes. By showing the positional extents of these temporal errors, we can investigate the interplay between velocity magnitude and time.

6.9 Evaluation and Results

We demonstrate these visualization techniques on a $[200 \times 200]$ subset of the top slice of a 3D simulation of global oceanic currents (see Data B.3.3), as shown in Figure 6.27. These simulations were produced using a technique based on a boundary impulse response functions [133] with the aim of studying the global eddies patterns in oceanic currents. The subset is mapped to a $[-1, 1] \times [-1, 1]$ domain. Although the flow in the original 3D simulation is incompressible, a 2D slice from this 3D flow does not satisfy incompressibility anymore, since slicing creates sinks and sources due to the flow across depths.

Figure 6.28(a) shows a streamwave at two error thresholds. The streamwave hits a saddle before exiting the domain from top-left. At high error, the streamwave flows into many slowly rotating critical points. A sudden and substantial increase in the thickness of the streamwave reveals the sensitivity of this region to the spatial error.

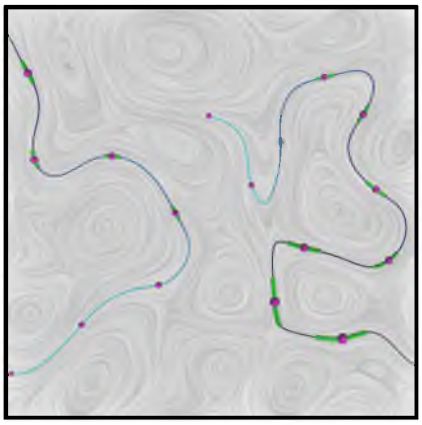


Figure 6.26. Temporal error in streamlines in the flow in the HCCI engine (see Data B.1.1). Two streamlines are seeded in the flow: one in the bottom left (exiting from top left), and the other in the upper center (exiting from the bottom right). Increasing time-steps are shown with increasing size of the magenta ball, and thickness of the green tube. With increasing time, the magenta ball travels along the streamline representing the movement of the particle, and the green tube elongates representing the increase in the spatial range due to the temporal error in the propagation.

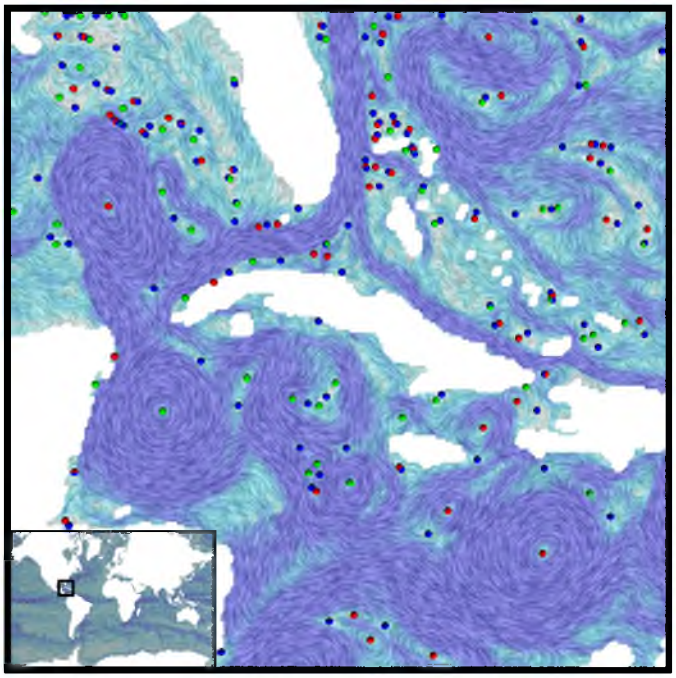


Figure 6.27. The oceanic currents in the Gulf of Mexico and the Caribbean Sea (see Data B.3.3). This [200 × 200] subset is taken from a larger simulation of oceanic currents [133] (inset), and mapped to a $[-1, 1] \times [-1, 1]$ plane. The region has 56 sources (green balls), 63 sinks (red balls), and 117 saddles (blue balls). The increasing speed of the flow is mapped to color from cyan to purple.

Figure 6.28(b) shows the spatial range corresponding to the temporal error as a sequence of time instants starting from 0 with a step of 0.3. A steady increase in the lengths of the green tubes is observed, reflecting a steady increase in the temporal error. This behavior is in conjunction with a steady increase in the thickness of the streamwave (Figure 6.28(a)) until it hits the saddle. Visualizing both these errors together is important to get a complete understanding of how the flow behaves under error.

Figure 6.28(c) shows the unstable manifolds grown from all the sources in the flow. These regions overlap with each other based on the approximation error in the flow. This overlap indicates the fuzziness in the boundary between them due to this error. Observe that higher refinement reduces the overlaps. There are many sources in the domain with very low divergence, in which case the corresponding manifold does not grow enough to be noticeable at this scale, especially at the higher refinement (bottom). Spatial refinement of Edge Maps increases their fidelity to piecewise linear flow, which can create cycles that were previously absent in the flow. Such cycles bound these growing regions. It is observed that large regions of the dataset remain untouched by these manifolds, suggesting that the

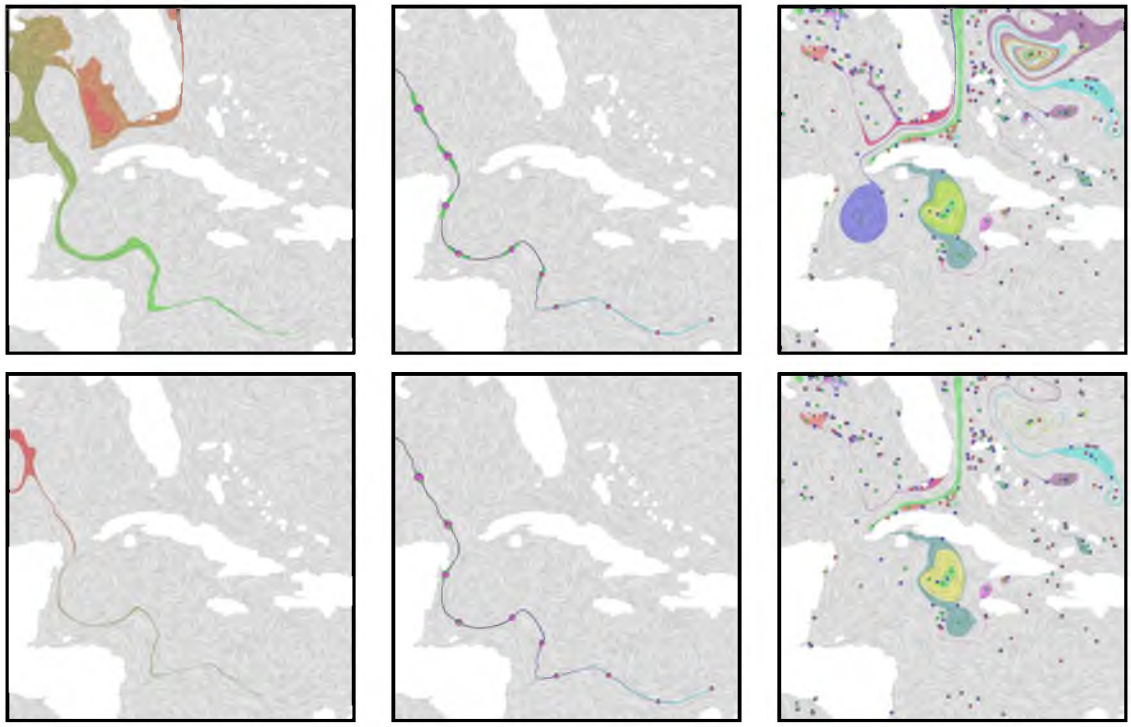


Figure 6.28. Streamwaves, temporal error, and fuzzy topology in the oceanic currents flow. (a) Streamwave visualization at a spatial error refinement of 0.1 (top) and 0.0001 (bottom). (b) Spatial range due to temporal error at a temporal refinement of 0.01 (top) and 0.0001 (bottom). (c) Unstable manifolds at a spatial refinement of 0.0001 (top) and 0.00001 (bottom).

flow in these regions is not associated with the sources in the domain. Hence, the flow in these untouched regions must either be bounded by orbits, or flowing in from the boundary of the domain.

The computation of Edge Maps for this dataset containing 63,010 triangles took 19.4 seconds and 19 MB, giving on an average 2.23 links per triangle. Spatial refinement (from unrefined maps) of 0.0001 took 29.5 minutes and 36 MB, giving 4.2 links per triangle, and temporal refinement (from unrefined maps) of 0.0001 took 25 minutes and 35 MB, giving 4.1 links per triangle.

6.10 Summary

This chapter presents Edge Maps, which is a new representation of piecewise linear vector fields. As compared to the traditional representation of such data that stores vectors at the vertices of a triangulation, Edge Maps encode the underlying flow (streamlines) directly in the representation. As a result, they remove the need for numerical interpolation and

numerical integration of vectors for streamline computation.

An attractive characteristic of Edge Maps is that they partition the flow through the triangle into connected pieces, and only a few special streamlines are needed to compute this partition, based on which they can represent the entire flow through the triangle. To compute these initial streamlines during preprocessing, a highly accurate analytic solution for linear vector fields can be used. It must be noted that although other techniques can be used for this purpose, Edge Maps will be only as accurate as these streamlines.

In order to concisely represent Edge Maps, base Edge Maps can be constructed as an approximation to the underlying Edge Maps. We use a linear interpolation of Edge Maps to compute the streamlines. However, unlike other representations and integration schemes, Edge Maps can capture this representation error (the error due to linear mapping). As a result, the error compounded during the streamline integration is not ignored, allowing for the visualization of the incurred error in both space and time. Furthermore, refinement of Edge Maps is possible to reduce this mapping error, and a user-defined threshold for error can be imposed on the system.

The most important property of Edge Maps is that they can preserve the order of streamlines through the triangle, and, therefore, can create consistent streamlines—the ones that never cross. The linear mapping discussed in this chapter respects this order preservation, but only up to floating-point precision. Due to the numerical nature of Edge Maps, there may still exist cases where consistency is violated. To fully utilize the theoretical advantages of Edge Maps, Chapter 7 presents an integer-based datastructure to store Edge Maps, which guarantees consistent streamlines.

In order to understand the construction and properties of Edge Maps, this chapter also discusses various properties of piecewise linear flow on triangulations, and also shows that there exist 23 equivalence classes of Edge Maps. Although the focus of the work presented here was piecewise linear flow on triangulated domains, the concept of Edge Maps as a boundary-to-boundary mapping is more general, and is applicable to other forms of interpolations and spatial discretizations as well.

For example, one could imagine Edge Maps for quadrilateral cells with bilinear interpolations. However, to construct such a representation, one needs to perform the corresponding study of equivalence classes and understand the properties of bilinear flow. Similarly, one can potentially create such representations for tetrahedral elements for 3D flow, which, however, appears very challenging, since instead of using subsets of edges, one must use arbitrarily-shaped subsets of the faces to map the flow. Another potential future direction is

creating Edge Maps for time-varying vector fields. Such representations must map pathlines instead of streamlines. However, this task also requires addressing multiple challenges, since each triangle will have an Edge Map for each time-step, and an efficient way of interpolating between these Edge Maps will be required.

CHAPTER 7

REPRESENTATION AND ANALYSIS OF FLOW WITH CONSISTENCY GUARANTEES

Computational models for experiments, simulations, and analysis are typically based upon *smooth theories*—mathematical theories that assume smooth fields defined over smooth domains. In practice, however, these tasks are performed using computing machines, which are fundamentally limited in how they can represent and process data. In particular, scientific data is represented numerically as sampled fields defined over discretized domains, and restricted to using floating-point numbers. Thus, although such floating-point representations are based upon smooth theories as well, they have a finite precision, and cannot represent the infinite-precision real numbers with 100% accuracy, causing representation errors. Furthermore, analysis techniques must incur rounding-off errors, and often make various numerical approximations while processing this data, thus adding to the error. Therefore, when *numerical analysis* based on a smooth theory is applied to sampled data, the obtained results contain substantial errors [70] (refer to the green column in Figure 7.1). This lack of numerical robustness can cause serious problems of *inaccuracy* and/or *inconsistency* (defined and discussed in Chapter 1).

In contrast to the numerical representations, *combinatorial approaches* based upon *discrete theories*—where both the domains and the functions are discrete—can be used to represent the data with full fidelity. Illustrated as the blue column in Figure 7.1, these representations, as well as the analysis they enable, typically use integers or graphs to represent and process the data, and do not require floating-point arithmetic. As a result, the addition of errors in the results can be controlled, and the underlying discrete theory can be replicated exactly in computer applications, enabling the analysis to produce accurate and consistent results using robust algorithms and datastructures. These discrete representations are useful for a variety of important phenomena such as graph traversals,

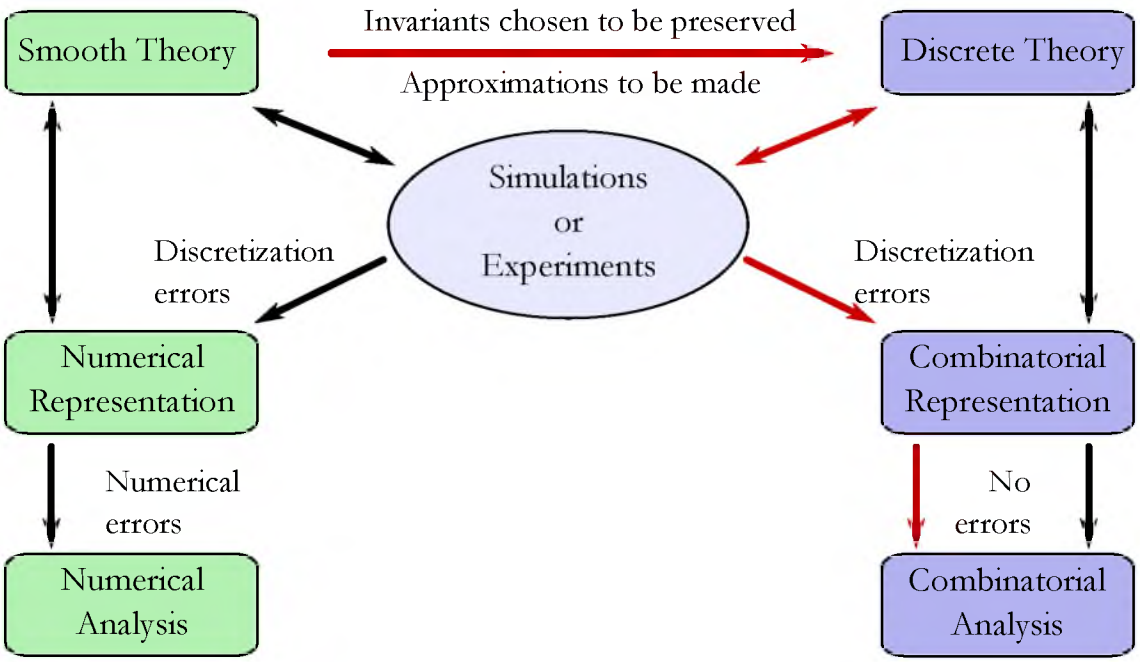


Figure 7.1. A schematic of two important paradigms of representations and analysis. Instead of smooth fields whose analysis represents the ground truth, numerical techniques must work with sampled data whose analysis can produce inaccurate as well as inconsistent results. Instead, combinatorial techniques, which can be used for discrete data only, offer significant advantages. This dissertation develops a new discrete theory for vector fields, with which the representation and analysis leads to accurate and consistent results.

number theory, binary logic, decision theory, etc.

The paradigm of discrete representations offers significant advantages over its numerical counterpart. However, vector fields—the type of data of interest in this dissertation—cannot be directly represented in discrete form as they capture smooth physical phenomena. In order to enable consistent feature extraction from vector fields, this dissertation underlines a shift in paradigm of representation and analysis of such data—in principle, also applicable to other forms of smooth data. In particular, for a given smooth theory, it is possible to create its discrete version, which can be implemented using the necessary combinatorial approaches; for example, Forman's discrete Morse theory [54, 57] provides an attractive counterpart to the Morse theory of smooth functions [137, 140]. Although the discrete theory makes certain approximations with respect to the smooth theory, the primary advantage is that it allows enforcing the physical laws and invariants that need to be preserved. Additionally, the loss of information during the conversion from smooth to discrete theory can be performed in a controlled manner, and the downstream analysis can be performed in an error-free manner. As a result, the overall error present in the pipeline

can be encoded and tracked.

In this context, the contributions of this dissertation are highlighted in red arrows in Figure 7.1. The work presented in this chapter develops the first comprehensive discrete theory for piecewise linear (PL) vector fields defined for triangulated domains. In particular, the goal is to preserve two important laws: (1) two streamlines do not intersect, and (2) the Poincaré index theorem is not violated. In the construction of this new theory, we make certain approximations; most importantly, we sacrifice the geometry of streamlines within each triangle. Nevertheless, as will be shown in this chapter, the corresponding combinatorial representation and algorithms produce consistent analysis—a significant advantage when deriving scientific insights from the data. The main characteristics of numerical and combinatorial approaches to representation and analysis are summarized in the following:

- The results of numerical analysis produce inaccurate and inconsistent results with respect to the smooth theory.
- The results of combinatorial analysis are both consistent and accurate with respect to the discrete theory.
- With respect to the smooth theory from which the discrete theory was developed, the results of combinatorial analysis are not 100% accurate, since some approximations have to be made during the initial conversion. Nevertheless, more importantly, the results are consistent with respect to the invariants that are preserved.

In this context, Edge Maps (presented in Chapter 6) can be seen as an intermediate paradigm (another vertical pipeline) between the smooth and the discrete theories. A comprehensive study of the theory of PL vector fields was presented (as equivalence of maps and the enumeration of possible cases), and a datastructure (Edge Map itself) was developed to implement this theory. Edge Maps enable encoding of discretization errors, because of a controlled conversion of PL theory to a numerical representation. Nevertheless, since Edge Maps are still numerical in nature, they could guarantee consistency only in theory (see Theorem 6.1). Guaranteeing consistency in practice requires developing and implementing a discrete theory, which will be presented next.

This chapter develops a new discrete theory for 2D piecewise linear vector fields. This theory defines critical points, streamlines, and periodic orbits that can be computed consistently. Extending the Edge Map datastructure presented in the previous chapter, this chapter presents *quantized Edge Maps*. Quantized Edge Maps use integers to represent the data instead of floating-point numbers, and can be used to implement the discrete theory developed above. Maintaining the error-encoding property of Edge Maps, their

quantized counterparts have an additional capability of representing the underlying flow (behavior of streamlines) as graphs to impart robustness and guarantee consistency. This chapter presents a new combinatorial framework for robust detection of critical points for 2D and 3D vector fields that fall under a large class of interpolation schemes, including the piecewise linear case. Together, these techniques provide a unified framework for consistent topological analysis of vector fields, as demonstrated in Section 7.3.

7.1 Consistent Streamline Tracing

To construct quantized Edge Maps for consistent streamline tracing, the framework from Edge Maps will be extended. Therefore, all notations, definitions, and assumptions used in this section are the same as described in Chapter 6 unless otherwise noted.

Edge Maps are maps between subsets of the triangle boundary ($\partial\Delta$), called *intervals*. Every point on $\partial\Delta$ is an inflow point, an outflow point, or a transition point. By definition, transition points, their images, and sepx points form the boundary of intervals in the map. Assumption 6.2 ensures that transition points on the edges of Δ are isolated, or conversely, no edge is entirely parallel to the flow. Furthermore, Assumption 6.1 disallows the existence of a critical point on $\partial\Delta$. This assumption is typically not satisfied in practice, and prevents the modeling of higher-order critical points. Therefore, for the construction of quantized Edge Maps, Assumption 6.1 is removed.

7.1.1 Quantized Edge Maps

To represent quantized Edge Maps, every edge in $\mathcal{M}$ is explicitly discretized into $n = 2^k$ bins, with $k = 32$ unless otherwise noted. A bin denoted as $\hat{\mathbf{p}}$ corresponds to the sample point $\mathbf{p}$ at its center, and, therefore, the terms *bins* and *samples* will be used interchangeably. A *quantized interval*, $\hat{\mathbf{P}}$, is a sequence of bins $\{\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_n\}$ along an edge of the triangle, and can be uniquely identified through two integers:

- its edge number (within the triangle under consideration), and
- its first bin (with respect to the edge under consideration).

Since intervals are stored contiguously, their extent can be inferred by their neighbor.

Given a pair of quantized origin and destination intervals, $\hat{\mathbf{O}}$ and $\hat{\mathbf{D}}$, a *quantized Edge Map* is defined as a map $\hat{\xi}\colon \hat{\mathbf{O}} \to \hat{\mathbf{D}}$, that, for an *inflow bin*, gives the destination of the streamline passing through it—the *outflow bin*. A quantized Edge Map is illustrated in Figure 7.2. Mapping an interval with m bins onto an interval with n bins can be solved by any rasterization procedure, such as Bresenham's algorithm [16]. Given a bin $i < m$,

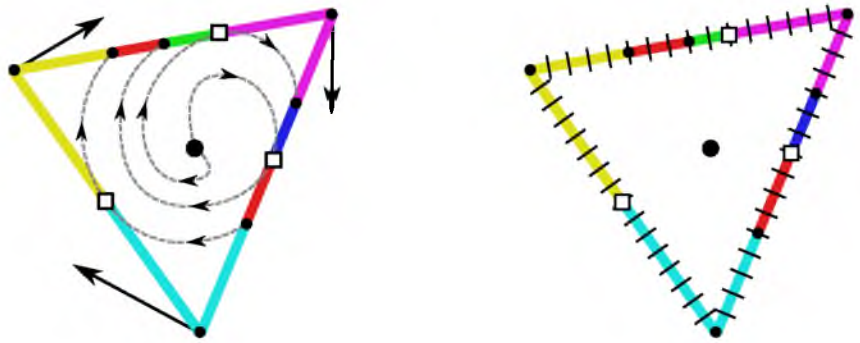


Figure 7.2. Quantized Edge Maps are created by discretizing the edges using $n = 2^k$ bins, and mapping intervals of bins that represent connected flow behavior. For clarity in presentation, the illustration uses only 16 bins per edge.

the destination of the streamline passing through it is computed by rounding $n(i/m)$ to the nearest integer. This computation can potentially produce a round-off error, in which case, the destination is taken to be $n(i/m) + 1$. The error introduced due to this linear mapping will be discussed in Section 7.1.1.5. It is important to note that although this rasterization emulates a linear interpolation, it is an entirely integer procedure, thus enabling consistent representation of flow.

7.1.1.1 Construction. One disadvantage of the original Edge Map representation is that it relies on constructing nondegenerate local flow numerically. Especially, the construction of Edge Maps requires tracing streamlines of transition points and saddle points—by definition, the most unstable structures in the flow. As a result, any numerical instabilities encountered during the construction can produce inconsistencies causing the construction of Edge Maps to fail. Instead, the quantized Edge Map construction is designed to, first and foremost, produce a consistent map independent of any errors in tracing streamlines, or classifying inflow/outflow points. Furthermore, contrary to the Edge Map construction (Algorithm 6.1), the structure of the mapping, that is, the number and connection of intervals, is computed using the most stable aspects of the flow and thus, typically produces a more accurate map as well. In particular, quantized Edge Maps are constructed using a divide-and-conquer approach for each triangle, where edges are divided into intervals where the flow can be mapped continuously, and the resulting intervals are paired to form links.

The first step of the construction algorithm identifies all transition points on the edges and vertices of a given mesh based on the interpolated vector field in each triangle, and are classified into internal and external. Since a triangle is convex, all transition points

on vertices must be external (see Lemma 6.2). Transition points on edges are classified by examining the local flow direction. Every bin in a quantized Edge Map is defined to be either an *inflow bin* or an *outflow bin*. Therefore, all transition points (and their images) must exist on the boundary of bins. At an ETP, the two bins on either side are mapped to each other, whereas at an ITP, both bins map to disparate locations. Furthermore, any ITP is immediately classified as ETP in its neighboring triangle since by definition, a dual ITP classification is inconsistent with a PL flow, since it creates a higher-order critical point. Unlike Edge Maps, this construction allows existence of critical points on edges, for example, a transition point classified as ETP in the triangles on both sides of the corresponding edge naturally produces a rotating critical point.

This classification is a combinatorial procedure, and can be performed robustly. However, it disregards the paths streamlines would take. If there exists a closed orbit in the interior of a triangle that touches an edge, the original Edge Maps would create a corresponding ITP (Figure 7.3(b)). However, according to the local flow behavior, it is classified as an ETP (Figure 7.3(c)). This discrepancy reflects the fact that quantized Edge Maps only store the flow behavior as it pertains to the boundary (the initial approximation made during the development of the corresponding discrete theory). Nevertheless, either classification is consistent with the given boundary map.

In the second step, an initial list of inflow and outflow intervals is identified based on the pre-identified transition points. Starting from the initial inflow/outflow intervals, the largest interval (inflow or outflow) is selected, and a streamline is traced (numerically using the local exact method [149]) from its center bin to identify its corresponding source/destination,

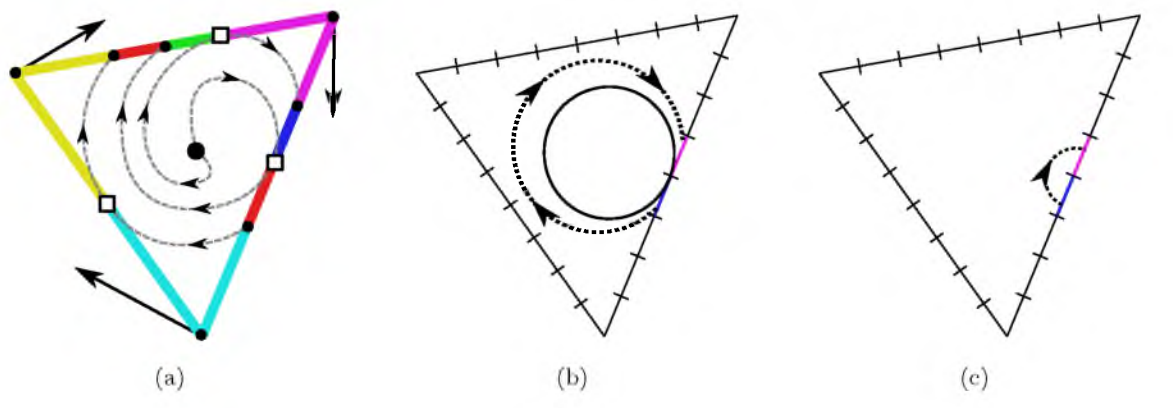


Figure 7.3. Combinatorial classification of transition points disregards the flow behavior inside the triangle. (a) Quantized Edge Map of a spiraling source. Transition points (white squares) and their images (black dots) form the colored intervals. (b) An interior orbit touching an edge creates an ITP whose boundary flow (c) appears as an ETP.

which divides the flow within the triangle into two subregions.

Subsequently, each region is handled recursively until one of the three possible final configurations is obtained: (1) a region without transition points, (2) a single ITP, and (3) a single ETP with only two intervals. In (1), there may exist either only two intervals that must be linked, or multiple intervals that must contain sepx points. For the latter case, candidates for separatrices and split the intervals are numerically identified. For (2), the forward and backward image of the ITP is identified and the intervals are split. Finally, for (3), the two intervals around the ETP are linked. By the time the algorithm resorts to numerically tracing potentially unstable streamlines, the basic structure of the maps is already established. Thus, the tracing simply computes the best geometric fit to a known flow structure. At last, a cleanup step is performed to merge intervals not separated by transition, image, or sepx points to construct the final quantized Edge Map.

7.1.1.2 Consistent flow representation. The construction of quantized Edge Maps relies on the tracing of streamlines, which can produce inconsistent results and numerical errors. This is, in fact, the primary motivation for constructing quantized Edge Maps. The algorithm described above, nevertheless, guarantees consistency by explicitly storing any numerical decision in the form of the region decomposition.

The order of intervals along the triangle boundary can be captured as a linked list, where each node of the list represents an interval, and a link between nodes represents that the corresponding intervals are contiguous. Since the triangle boundary is closed, the initial linked list is circular. The first streamline that is traced divides the triangle into subregions, such that no subsequent streamline can leave its subregion. This subdivision is equivalent to the splitting of the initial circular linked list into two noncircular linked lists. Each region (and the corresponding linked list) represents flow that must stay within itself. Therefore, any subsequent streamline tracing forces the source and destination bins to be a part of the same linked list. As a result, it becomes possible to prevent any streamline crossings, thereby guaranteeing a consistent representation of the flow. Note that this approach does not prevent numerical errors, but instead forces any numerical decision to defer to all previous decisions.

7.1.1.3 Datastructure. For each triangle, a circular linked list of intervals around the boundary is stored. The datastructure to store each interval in a quantized Edge Map is illustrated in Figure 7.4. Each interval stores a k-bit integer as the index of the first bin for the interval. Additionally, two bytes per interval are bit-packed with the following information: 2-bits for the edge number within the face since bin numbering is only

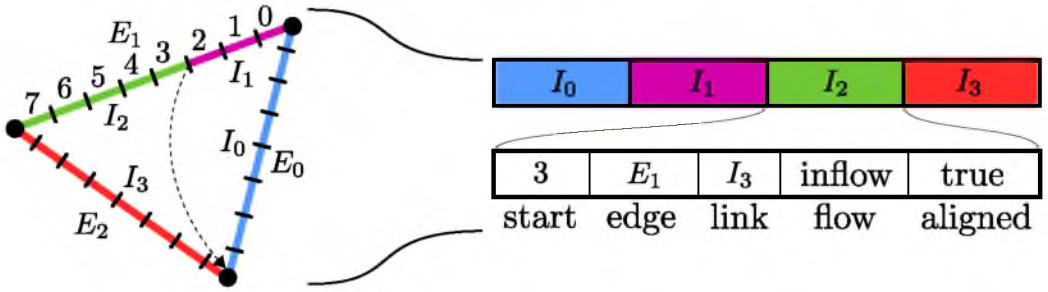


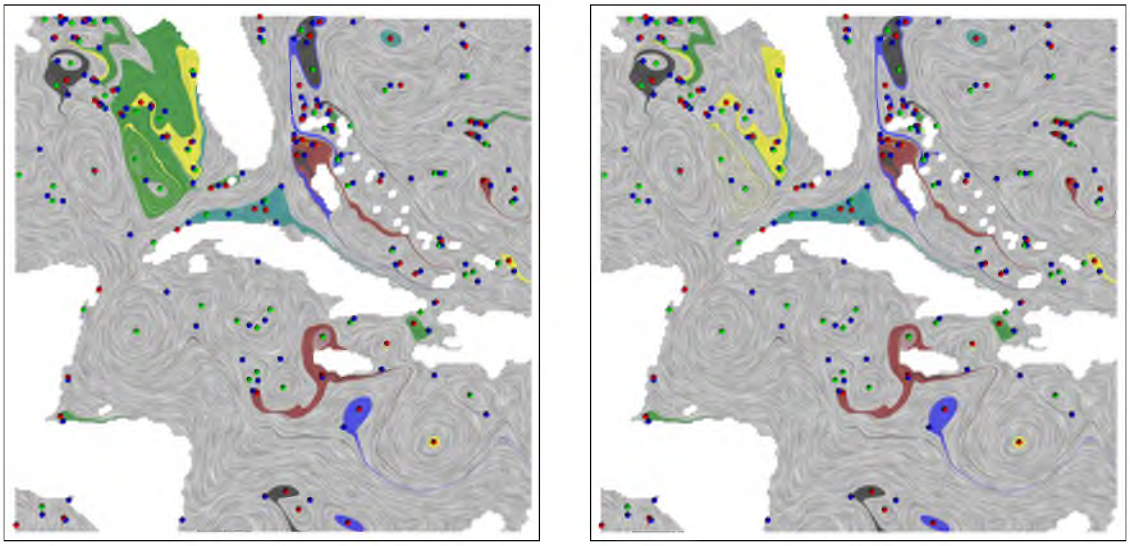
Figure 7.4. The quantized Edge Map of a triangle is stored as a counterclockwise oriented linked list of intervals. Each interval stores a k-bit value ($k = 3$ here) for the beginning bin of the interval and a two-byte descriptor.

unique per edge; a 1-bit flag indicating inflow or outflow; 1-bit indicating the orientation of the interval, relative to the bin numbering; and 12-bits for the index of the paired source/destination interval.

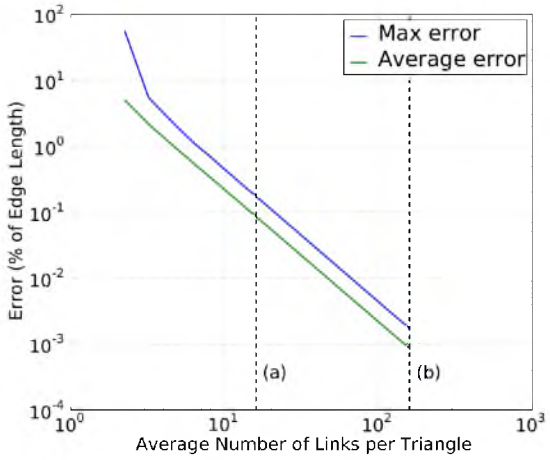
7.1.1.4 Quantization error. The conversion of edges into a sequence of bins can cause quantization error, which is controlled by the number (and, hence, size) of bins. Varying the number of bins provides variable granularity or precision in representing flow, and determines how many unique streamlines are used to represent the flow. The geometric location of all landmarks, that is, critical points, transition points, images points, etc., can be off by at most half the bin size. In practice, using 2^{32} bins makes the precision of this representation higher than possible with a 32-bit floating-point representation. Furthermore, these bins are distributed uniformly along edges, unlike the IEEE floating-point standard. Thus, when compared to vertex-sampled and interpolated flows, the quantization error is negligible, especially considering the gained consistency.

7.1.1.5 Linear mapping error. The error introduced due to linear mapping between intervals, on the other hand, cannot be neglected. Similar to that in the Edge Maps, the mapping error of the quantized Edge Maps at a given source bin $\hat{\mathbf{p}}$ is represented in terms of the distance between its mapped destination $\hat{\xi}^{+}(\hat{\mathbf{p}})$ and its true destination relative to the edge length. Unfortunately, there exists no closed-form solution for this error, but experimentally (evaluated using the local exact method [149]), it appears to be smooth and well behaved. Thus, it is possible to evaluate the error for each link using a dense sampling. Furthermore, similar to Edge Maps, their quantized counterparts can also be refined by selecting links with high error, and iteratively splitting them using the streamline at the point of maximal error. Assuming fewer than 2^{12} intervals per triangle (the interval array is indexed using 12 bits), each split requires only $2(32 + 16)$ additional bits of storage (the overhead for storing two new intervals). Although refinement requires careful sampling,

after map refinement the extra intervals do not have much effect on the runtime for map queries. Figure 7.5 shows the effect of refinement on the stable and unstable manifolds of the 2D global oceanic currents (see Data B.3.3), and the observed convergence in terms of the maximal and average error for a given number of links. It can be seen that this error is well behaved in both a maximal and average sense, with only a small initial dip as flow is refined (mostly near saddles) initially.



(a) (b)



(c)

Figure 7.5. Refinement and error-convergence of quantized Edge Maps. The refinement of links at the point of maximal error from (a) 0.18% to (b) 0.023% of the edge length shows a large change in the unstable region to the west of Florida (top-left of the flow). (c) Total mapping error as a function of average number of links shows well-behaved convergence as more links are used.

7.1.2 Quantized Streamlines

Quantized Edge Maps provide a consistent representation of the flow within each triangle. This section describes how they can be used for tracing consistent streamlines across multiple triangles. In particular, the quantized Edge Maps allow finding the destination bin of a given sample point (bin) robustly using integer arithmetic. Therefore, the streamlines traced using quantized Edge Maps are discrete sets bins on the edges of triangles reached by following the destination maps forward and the source maps backward.

Definition 7.1 (Quantized Streamline)**.** A *forward quantized streamline*, $\hat{\mathbf{x}}_S^+(\hat{\mathbf{p}}_0)$ starting at bin $\hat{\mathbf{p}}_0$ is an ordered sequence of bins $\{\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_n\}$ such that $\hat{\xi}^+(\hat{\mathbf{p}}_i) = \hat{\mathbf{p}}_{i+1}$. A *backward quantized streamline* $\hat{\mathbf{x}}_S^-(\hat{\mathbf{p}}_0)$ is defined symmetrically using $\hat{\xi}^-$.

A quantized streamline disregards the path of the true streamline, and instead stores a pair of bins $(\hat{\mathbf{p}}_i, \hat{\mathbf{p}}_j)$. These points are called *source* and *destination* points, and the pair is called a *source-destination pair*. However, in order to use quantized streamlines as a stand-in for smooth streamlines, it must be ensured that they respect the same set of invariants—most importantly the fact that streamlines do not intersect. Using only the above definition of source and destination does not necessarily preserve this invariant. Although each sample has exactly one source and one destination, multiple samples can have the same source/destination and the source-destination maps are not inverse to each other. Therefore, in areas of sufficient divergence, quantized forward streamlines can cross backward lines, creating an inconsistent flow (see Figure 7.6(a)).

To avoid this problem, the mapping can be modified slightly to define the forward map $\hat{\xi}^+(\hat{\mathbf{p}})$ of a sample $\hat{\mathbf{p}}$ by considering all pairs $(\hat{\mathbf{p}}, \hat{\mathbf{q}}_i)$. In particular, $\hat{\xi}^+(\hat{\mathbf{p}})$ is defined as the

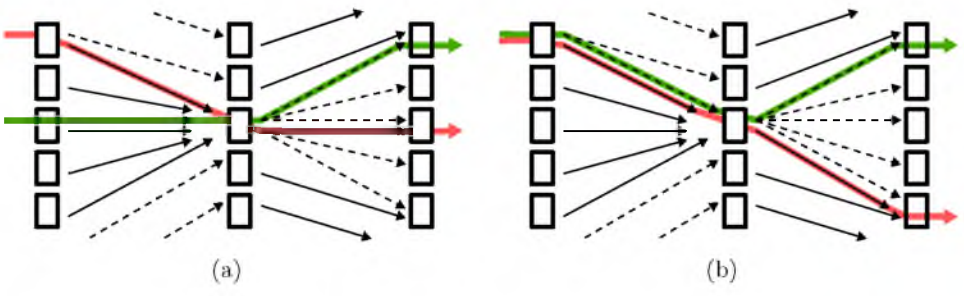


Figure 7.6. Quantized streamlines using forward and backward quantized Edge Maps are indicated by the solid and dashed arrows, respectively. (a) The red streamline (using $\hat{\xi}^+$) intersects the green streamline (using $\hat{\xi}^-$). (b) Adjusted $\hat{\xi}^\pm$ pairing picks the rightmost source/destination with respect to the tracing direction. The forward and backward streamlines may share a bin but do not intersect.

rightmost (relative to $\vec{V}$) sample $\hat{\mathbf{q}}_i$. Symmetrically, the backward map $\hat{\xi}^-(\hat{\mathbf{p}})$ is defined as the leftmost sample among all pairs $(\hat{\mathbf{q}}_i, \hat{\mathbf{p}})$; see Figure 7.6(b). Using the modified maps ensures that quantized streamlines may share samples (equivalent to becoming infinitely close) but never cross, thus, building quantized streamlines that are a provably consistent approximation of the flow.

7.1.3 Quantized Streamlines as Graphs

An important advantage of the quantized flow is that it can be represented as a multiresolution graph. In particular, since two triangles sharing an edge also share the bins on that edge, the maps of such triangles represent inflow and outflow consistently across their shared edges. As a result, the maps across the triangle edges can be connected to form a large graph called the *bin-graph*. Separate bin-graphs are needed to represent forward and backward flows. The forward and backward bin-graphs have the same nodes (bins) but different connections (mapping) between them.

Definition 7.2 (Bin-Graph)**.** Let $\hat{\mathbf{P}}$ be the set of bins on a triangulation. The *forward bin-graph* is a directed graph $\mathsf{G_B}^+(\hat{\mathbf{P}}, \mathsf{E}^+)$, such that

- the nodes are the set of bins on the given triangulation, $\hat{\mathbf{P}}$.
- the directed edges are the pairs of all source/destination bins under the quantized Edge Maps of all triangles, that is, $\mathsf{E}^+ = \{(\hat{\mathbf{p}}, \hat{\xi}^+(\hat{\mathbf{p}})) \mid \hat{\mathbf{p}} \in \hat{\mathbf{P}}\}$.

The *backward bin-graph*, $\mathsf{G_B}^-$, is defined symmetrically using the backward map, $\hat{\xi}^-$.

Corresponding to the quantized Edge Maps for the triangles shown in Figure 7.7, the forward and backward bin-graphs are shown in Figure 7.8. Using the bin-graphs, the flow can be represented as a finite number of streamlines possible through the traversals through this graph starting from any given bin. Each bin uniquely represents one forward and one backward streamline. Together, forward and backward bin-graphs describe the complete flow in a combinatorial manner. However, since they capture every bin in the triangulation as a separate node, their size is prohibitively large for practical applications.

Instead, a coarser representation of the vector field called *link-graph* can be defined for practical use. To define the link-graph, it is important to realize that converting quantized flow into graphs requires two types of mappings: the mapping within each triangle (links of quantized Edge Maps), and the mapping (across edges) between neighbor triangles. The bin-graphs directly encode the former, and the latter is a trivial one-to-one mapping since neighboring triangles share bins. However, when considering links defined as pairs of

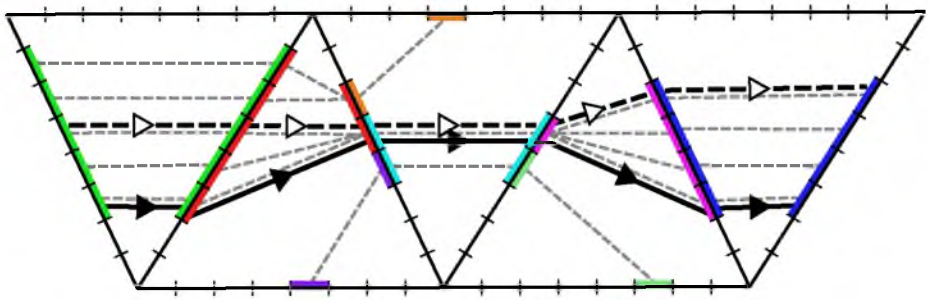


Figure 7.7. Selected links of the quantized Edge Maps for a set of triangles are shown in different colors, with grey dashed lines for each bin-to-bin rasterization. Two quantized streamlines are shown: forward with a solid black line and black triangles; and backward with a dashed black line and white triangles. Triangle direction indicates the direction of the vector field, not the direction of integration. In both cases, the quantized streamlines choose the rightmost (with respect to the integration direction) bin.

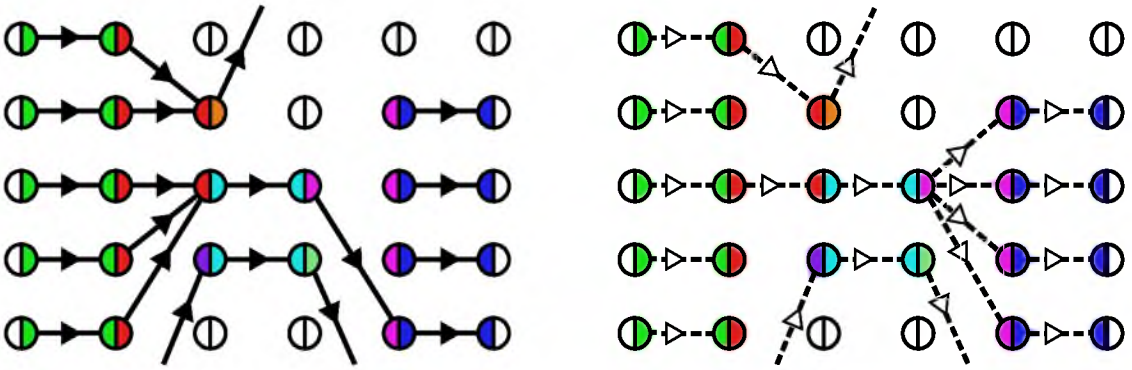


Figure 7.8. Forward and backward bin-graphs for the quantized flow in Figure 7.7. For each edge in this sequence, the bins are displayed as a column of circles, colored to indicate the links they fall in on either side of the edge.

sequences of source and destination bins, the intratriangle map becomes implicit, whereas the intertriangle map is nontrivial.

Consider the bin-graphs shown in Figure 7.8. The intratriangle mappings are indicated by the arrows, whereas the intertriangle map is implicit in the ordering of bins. Figure 7.9 shows the corresponding link-graph, where the nodes (boxes) represent the set of bins indicated by the color, and the edges encode that links have overlapping sequences of bins across an edge. For example, the red link's destination bins are split between the blue and orange link and, thus, the red node has two directed edges to the blue and orange nodes, respectively. Thus, the link-graph stores the intertriangle maps explicitly (as edges), whereas the intratriangle map is maintained internally within each node as a rasterization between two sequences of bins.

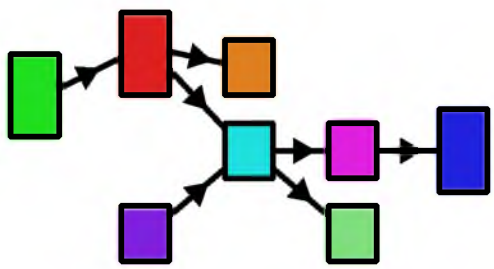


Figure 7.9. Link-graphs for the quantized flow in Figure 7.7.

Definition 7.3 (Link-Graph). A *link-graph* is a directed graph $\mathsf{G_L}(\mathsf{V_L}, \mathsf{E_L})$, such that

- $\mathsf{V_L} = \{\eta_i\}$, where each link-node, η_i, represents a contiguous sequence of bins from a single link.
- $\mathsf{E_L} = (\eta_1, \eta_2)$, such that η_1 and η_2 share at least one bin.

Since the link-graph, by definition, ignores the intratriangle mapping, it represents paths between links for which no quantized streamline exists. For example, the dark green and light green links in Figure 7.9 are connected even though no quantized streamline exists that shares both these bins (see Figure 7.7). However, the link-graph is a conservative representation, meaning that if two links share a quantized streamline, they are connected in the link-graph, but the reverse is not necessarily true.

7.1.4 Quantized Critical Points

In PL vector fields, critical points are typically detected by numerically solving for points with $\vec{V}(\mathbf{x}) = \vec{0}$, and classified using the eigenvalues of the Jacobian matrix of $\vec{V}$ [89]. However, this classification is limited to nondegenerate first-order critical points. Furthermore, finding exact zeros is numerically unstable. An alternate classification is to use the Poincaré index [92], which captures the aggregate structure of $\vec{V}$ on the interior of a closed curve γ. In particular, if γ encloses no critical points, then $\text{ind}_\gamma(\vec{V}) = 0$, and if there exists only a single critical point, the $\text{ind}_\gamma(\vec{V})$ is called the index of the critical point. Finally, given a set of critical points $\mathbf{c}_i$ with a simple enclosing curve γ, the Poincaré index, $\text{ind}_\gamma(\vec{V}) = \sum_i \text{ind}(\mathbf{c}_i)$, where $\text{ind}(\mathbf{c}_i)$ is the index of the critical point $\mathbf{c}_i$. The Poincaré index can distinguish isolated higher-order critical points such as k-saddles with $\text{ind}_\gamma(\vec{V}) = -k$ or k-poles with $\text{ind}_\gamma(\vec{V}) = +k$ when they are enclosed by γ.

Nevertheless, computing the Poincaré index in practice is nontrivial as it becomes difficult to ensure that γ does not contain a critical point. Instead, using quantized flow, ind_γ can be computed by a simple count of transition points. In particular, one can imagine transforming $\vec{V}$ in an ϵ-neighborhood of γ such that $\vec{V}$ is orthogonal to $\mathrm{d}\gamma/\,\mathrm{d}t$, and switches infinitely fast from outflow to inflow at transition points, thus, concentrating the rotation of $\vec{V}$ at the vertices and transition points and does not affect ind_γ. This process can be seen as the equivalent of using exterior angles as a discrete curvature measure for polygonal curves [72], where all curvature is concentrated at vertices. In particular, for a counterclockwise traversal around a triangle Δ, each transition point contributes a rotation of π (positive for ITPs and negative for ETPs), and each vertex contributes α, its exterior angle to ind_γ (which sum to 2π for all three vertices). When there is an ETP on a vertex, it is possible to separate the contribution of the transition point from that of the exterior angle. Thus, it is possible to compute the Poincaré index using just the number of ITPs and ETPs in a triangle.

$$\text{ind}_{\partial\Delta} = \frac{1}{2\pi}\left(\sum_{\|\mathbf{I}\|}\pi + \sum_{\|\mathbf{E}\|}-\pi + 2\pi\right) = (\|\mathbf{I}\| - \|\mathbf{E}\|)/2 + 1, \tag{7.1}$$

where $\mathbf{I}$ and $\mathbf{E}$ are the sets of ITPs and ETPs, respectively. Note that the Expression (7.1) ignores potential critical points that may exist at the edges or the vertices of Δ. Since the classification of transition points into ETPs and ITPs is made based on the local map, critical points on the boundary of a triangle can be treated as conventional transition points, effectively computing the Poincaré index of $\mathring{\Delta}$. Computing the Poincaré index for all triangles detects and classifies all critical points on the interior of triangles.

For critical points on edges, one can compute ind_γ, or the boundary of the quadrilateral formed by the adjacent triangles, and subtract the index of both triangle interiors, which is possible since transition points at quadrilateral corners can be easily classified into ETPs and ITPs based on the local maps. Note that the algorithm of Section 7.1.1.1 avoids creating critical points on edges, but other approaches may not, and the representation naturally supports edge criticalities. Finally, the Poincaré index of a vertex is computed using the boundaries of its star and subtracting the triangle and edge indices. Overall, this strategy provides a generic, combinatorial detection and classification of critical points, including the often problematic cases of critical points near vertices and edges.

7.1.5 Quantized Closed Orbits

An equally important aspect of vector field analysis focuses on other invariant structures than just critical points. A general vector field can contain closed orbits that generically can be attracting, repelling, or both. Closed orbits are particularly difficult to detect with numerical techniques. Instead, a quantized flow is, at its core, described by a very large graph, and thus, closed orbits exist explicitly as circular quantized streamlines.

Definition 7.4 (Quantized Cycle)**.** A *quantized cycle* is a quantized streamline $\hat{\mathbf{x}}_S = \{\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_n\}$, such that $\hat{\mathbf{p}}_0 = \hat{\mathbf{p}}_n$ for some $n > 0$.

Since the forward and backward maps are different, there exist three types of closed orbits: those stable in $\hat{\xi}^+$, those stable in $\hat{\xi}^-$, and those stable in both directions. Also, closed streamlines are limit sets and can be classified based on whether they are attracting, repelling, or neither on the sides of their trajectories. Furthermore, due to some subtleties in the definition of the the maps $\hat{\xi}^\pm$, there are only six classifications.

Consider the forward-stable closed orbit $\mathbf{C}$ shown in Figure 7.10. Since $\mathbf{C}$ is defined by $\hat{\xi}^+$ (shown as solid arrows), there cannot exist a streamline diverging from $\mathbf{C}$ towards its right (with respect to the flow direction). Thus, on its right, $\mathbf{C}$ can either be attracting (Figure 7.10(a))—if there exists at least one other streamline converging to it—or neutral (neither attracting nor repelling). On its left, $\mathbf{C}$ can be neutral if no other streamline approaches, attracting if there exists one or more forward streamlines approaching $\mathbf{C}$ (Figure 7.10(c)), or repelling otherwise (Figure 7.10(b)). The reason for this classification is that if there exists one or more forward streamlines that approach $\mathbf{C}$ from the left, then all backward streamlines that share bins with $\mathbf{C}$ will ultimately leave $\mathbf{C}$. However, all

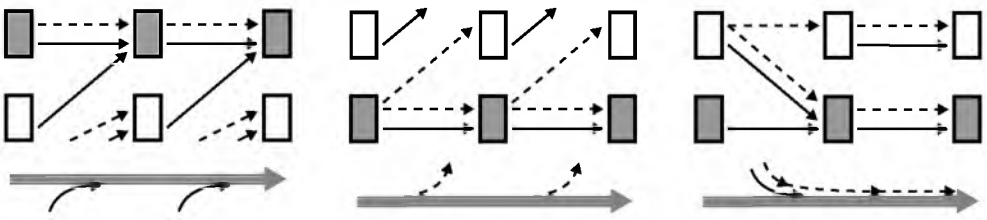


Figure 7.10. Classifications of forward-stable closed orbits shown as grey bins. Forward maps/streamlines are indicated by solid arrows and backward by dashed arrows. (a) On the right of the closed orbit, streamlines can only converge leading to attracting behavior. (b) If no forward streamlines approach from the left, the closed orbit is repelling from the left. (c) If a single forward streamline merges with the closed orbit, it is attracting; all backward lines only touch it.

streamlines are considered as distinct entities, which may become infinitely close but never touch. Such backward streamlines are never part of **C**, but simply approach and leave its vicinity. As a result, a forward-stable closed orbit can be attracting, repelling, or neutral on its left, but only attracting or neutral on its right. Symmetrically, a backward closed orbit can be attracting, repelling, or neutral on its left but only repelling or neutral on its right. One consequence of these properties is that no quantized closed orbit can switch between attracting and repelling, and thus cannot form a saddle-like region in the sense of the Conley theory [27].

7.1.5.1 Exact detection of closed orbits. To detect all closed orbits in a flow, a two-stage process is used. First, the link graph of the given flow is created. The link graph is a conservative description of the flow, such that two links sharing a streamline are connected in the link graph, but not all connected links share a streamline (see Figure 7.11). Subsequently, all maximally-strongly-connected components (MSCCs) are extracted from this graph using the algorithm of Barnat and Moravec [9]. Since usually most triangles are noncritical, they contain at most four links, and thus the initial link graph typically has four links per triangle on average. However, since all critical points have already been identified, one can avoid creating any nodes whose flow is entirely connected to sinks or sources, which drastically reduces the size of the initial link graph, and also splits it into independent components that can be processed separately.

After the first stage, each MSCC represents a set of bins forming a region that may contain closed orbits. To identify individual closed orbits, the initial graph is pruned by geometrically shrinking its nodes. The region is shrunk by removing the bins that either enter or leave the region of interest. Once no more bins can be pruned, the boundary

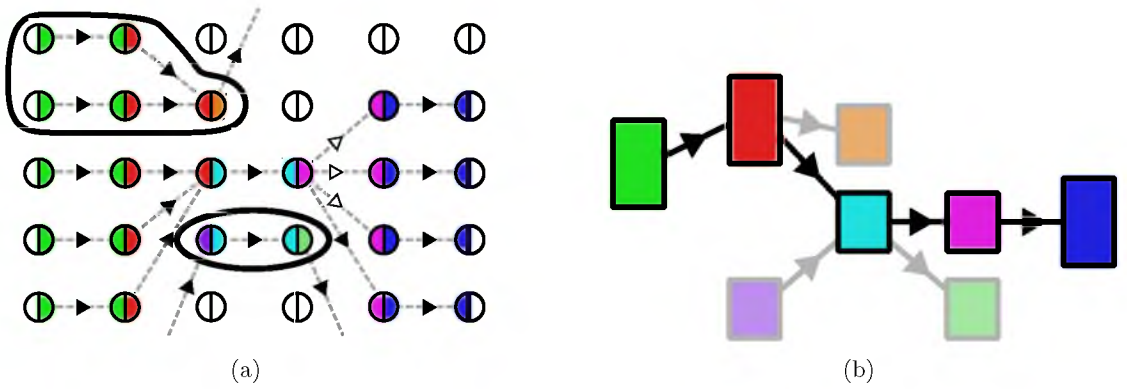


Figure 7.11. Exact closed orbit detection. The bin-level connectivity (a) defines the initial link graph (b). Extracting MSCCs removes the faded links, and the bin-level pruning removes the circled bins.

of the MSCC forms a bin-level closed orbit. Next, all the bins in the closed orbit are removed, exposing new bins to the pruning. This process continues until all regions are empty. A closed orbit stable in one direction may not be stable in the other. As a result, pruning bins according to $\hat{\xi}^+$ may remove parts of a closed orbit in $\hat{\xi}^-$ and vice versa. Thus, pruning is performed twice—individually to identify all forward and backward closed orbits. Figure 7.12 shows all closed orbits in the 2D global oceanic currents (see Data B.3.3) around Italy with the color indicating the classification.

7.1.5.2 Approximate closed orbit detection. The pruning algorithm is guaranteed to find all closed orbits as a sequence of bins that can subsequently be classified. However, in practice, there can exist regions with extremely low divergence, and in many applications, vector fields are theoretically divergence-free. Typically, these are not represented exactly and the resulting approximated vector field has near-zero divergence. In such cases, a quantized flow may represent billions of closed orbits. Extracting these individually is neither practical nor desirable. Instead, potential closed orbits can be approximated up to a user threshold.

To approximate closed orbits, MSCCs are split by tracing quantized streamlines. Specifically, the MSCC node with the largest width of bins is selected, and forward and backward streamlines are traced through the bin at the center of the corresponding link. For all nodes that are part of these streamlines, the corresponding bin is duplicated, the node is split into two disconnected nodes, and arcs of either node are updated. The streamlines can: (1) form a closed orbit, (2) leave the MSCC, or (3) connect to a source and/or sink, all of which significantly change the connectivity of the graph. To avoid tracing such a streamline for too long, a user-specified maximal length is used, after which the tracing is terminated. Note that even a partial streamline splits graph nodes and thus provides

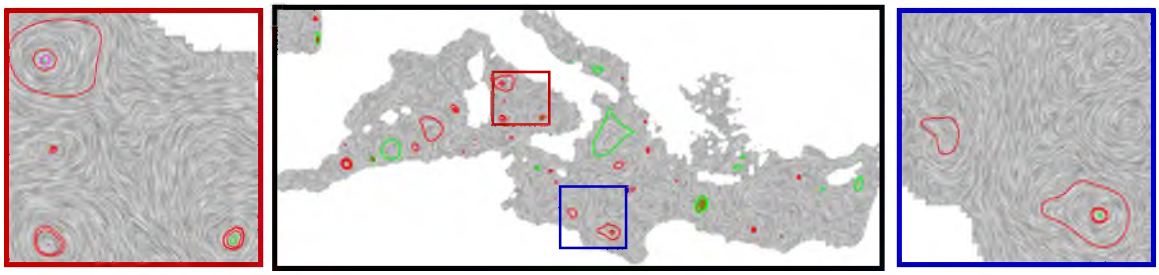


Figure 7.12. Classification of closed orbits in the 2D global oceanic currents (see Data B.3.3) around Italy. The flow is dominated by both-side attracting (red) and both-side repelling (green) closed orbits. Blue closed orbits are neutral on both sides, and are both forward- and backward-stable. Other types of closed orbits exist but are too small to be seen.

new information. Once the graph connectivity has been updated, new MSCCs are looked for. This process is iterated until the cross-section of all remaining regions falls below a threshold of minimal number of bins, as shown in Figure 7.13. These regions become the current approximation of potential closed orbits, and their aggregated flow behavior can be determined by examining the maps along their boundary.

7.1.5.3 Comparison. Figure 7.14 presents a comparison of the closed orbit detection technique presented in this chapter with that of Chen et al. [28] and Szymczak and Zhang [198]. The flow representing the 2D global oceanic currents (see Data B.3.3) in a small region around Iceland containing 23,655 triangles is used for this purpose. This figure highlights differences between the presented technique and the other two, in part because of the instabilities and very small scale orbits that are contained within. The triangle-based approach of Chen et al. identifies 160 Morse sets. Szymczak and Zhang's approach detects 297 Morse sets, of which 113 are trivial. In this example, they have refined their transition graph with 11 iterations of refinement, subdividing far beyond the Morse sets seen by Chen et al. Our approach detects 122 closed orbits and 166 critical points (45 sinks, 38 sources, and 83 saddles) after refining to a threshold of 0.023%.

The limit sets that each technique produces are qualitatively quite similar, as depicted for the larger orbits. This example shows that numerical instabilities are handled differently by each approach. Chen et al. prefer a coarser approximation, covering whole triangles with regions that behave as Morse sets. Szymczak and Zhang find a finer scale approximation by refining the transition graph represented by their PC flow. One difference with our technique is explained in the blue box of Figure 7.14. Our refinement, instead of equally splitting links as done by Szymczak and Zhang, splits links based on an error measure,

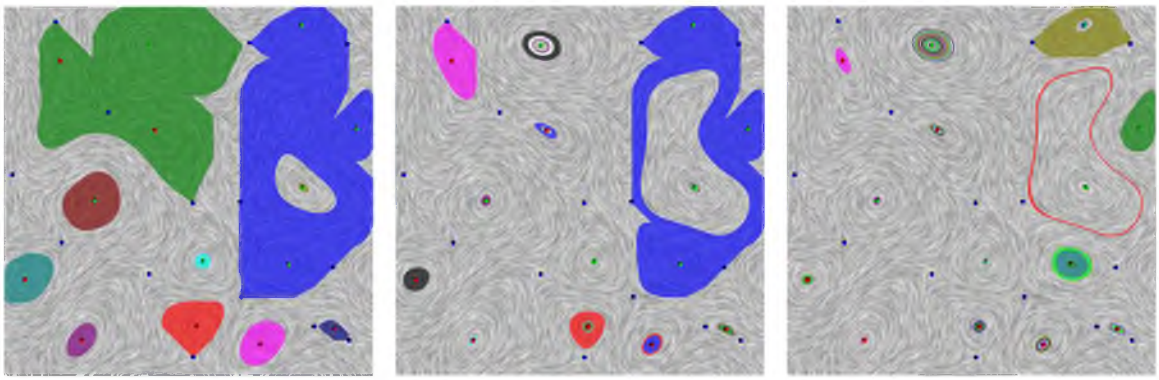


Figure 7.13. Graph-splitting progressively converges to closed orbits in the flow inside the HCCI engine (see Data B.1.1). (Left-to-right) A gradual reduction in the size of regions.

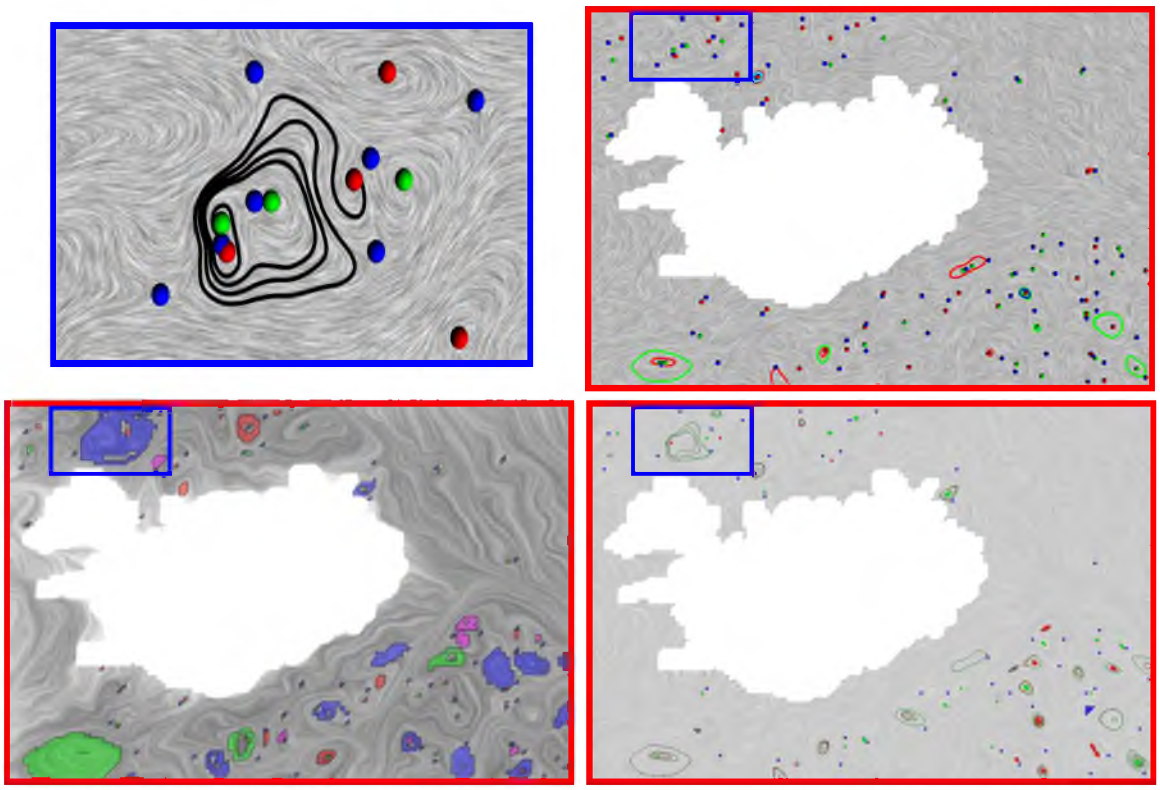


Figure 7.14. Comparison of closed orbit detection with Morse sets (bottom-left, Chen et al. [28]), and refined PC Morse sets (bottom-right, Szymczak and Zhang [198]). Both Morse techniques compute cyclic regions in the blue boxed region, but our approach (top-right) identifies none, as the visualization in the top-left confirms by indicating a streamline that spans the region.

which indicates geometric closeness to the original PL flow. It turns out that in this region, we find no closed orbits, because the PL flow prescribes a streamline (computed using the local exact method [149]) that crosses over the closed orbit regions detected by the other two techniques, which leads to an interesting contrast between the three discrete approaches. All three approaches make some sacrifice when computing the input PL data to a combinatorial representation. However, our approach strives to capture as much geometric information as possible by encoding an approximation of the streamlines prescribed by the input PL field.

7.2 Consistent Critical Point Detection

As discussed in Chapter 1, consistent critical point detection is also a key component for reliable analysis of vector fields. A manifestation of inconsistency in critical point detection was illustrated in Figure 1.5. To address such problems, this section presents a solution to guarantee consistent critical point detection. In particular, this section makes the following contributions:

1. It proves necessary and sufficient conditions for the existence of critical points within cells of a simplicial mesh (in both 2D and 3D) for a broad class of interpolation functions, and
2. It shows how to turn these necessary and sufficient conditions for the existence of singularities into a combinatorial algorithmic approach.

7.2.1 Existence of a Critical Point in a Simplex

The discussion presented in this section assumes the availability of a generic sampled vector field (see Definition 2.10). This assumption simplifies the underlying theory required to develop the technique for critical point detection. Later, Section 7.2.2 discusses how this assumption can be relaxed.

In the following expressions, $\mathcal{S}$ represents a simplex, $\mathring{\mathcal{S}}$ the interior of the simplex, and $\mathcal{M}$ a simplicial complex. The star of a given vertex $\mathbf{x}$ is denoted as $\mathrm{St}(\mathbf{x})$. This work focuses on a class of interpolated vector fields that can be expressed as

$$\begin{aligned} \vec{V}(\mathbf{x}) &= \sum_{\mathring{\mathcal{S}}_i \in \mathcal{M}} \vec{V}_i(\mathbf{x}), \\ \vec{V}_i(\mathbf{x}) &= \sum_{\mathbf{x}_j \in \mathcal{S}_i} \alpha_j(\mathbf{x}) \vec{V}_j, \end{aligned} \tag{7.2}$$

where the weight functions $\alpha_j(\mathbf{x})$ defined for vertices $\mathbf{x}_j$ are continuous, non-negative, and local, meaning $\alpha_j(\mathbf{x}) > 0, \forall \mathbf{x} \in \mathrm{St}(\mathbf{x}_j)$, and $\alpha_j(\mathbf{x}) = 0, \forall \mathbf{x} \notin \mathrm{St}(\mathbf{x}_j)$.

Following the definition of the weight functions $\alpha_j(\mathbf{x})$, it is clear that for the simplex $\mathcal{S}_i = \{\mathbf{x}_{i_0}, \ldots, \mathbf{x}_{i_d}\}$, $\alpha_j(\mathbf{x}) > 0$ only for $j \in \{i_0, \ldots, i_d\}$. Furthermore, $\alpha_j(\mathbf{x}) \to 0$ as $\mathbf{x} \to \mathcal{S}_j$, where $\mathcal{S}_j$ is a facet of $\mathcal{S}_i$. Since vector samples, $\vec{V}_i$, are defined only on the interior of simplices, the resulting vector field, $\vec{V}$, is C^0 continuous across the faces of the simplices. It is simple to confirm that PL interpolation falls into this class of functions. Also, a variant of Radial Basis Function (RBF) interpolation falls into this class where the weights smoothly fall to zero at the boundary of the vertex stars.

Using the concepts introduced in Section 2.2, this section will show that a simplex $\mathcal{S}$ contains a critical point if and only if the origin, $\mathbf{0}$, lies in the convex hull of the vectors at the simplex's vertices. To connect the results of the Brouwer degree with vector fields on simplices, a one-to-one mapping from a simplex to an enclosing ball is defined. Let $\mathcal{S} = \{\mathbf{x}_0, \ldots, \mathbf{x}_m\}, m \leq n, \mathbf{x}_i \in \mathbb{R}^n$ be an m-simplex of $\mathcal{M}$, and $\mathbb{B}$ be a m-dimensional unit ball, that is, $\mathbb{B} = \{\mathbf{x} \in \mathbb{R}^m \mid \|\mathbf{x}\| = 1\}$. Without loss of generality, assume that the origin $\mathbf{0}$ belongs to the interior of $\mathcal{S}$. Let $\mathbf{x} (\neq \mathbf{0})$ be a point in the interior of $\mathcal{S}$, and $\mathbf{x}'$ be the intersection of the boundary of $\mathcal{S}$ with the ray from the origin through $\mathbf{x}$. Then, define

a mapping $\mathfrak{B} : \mathcal{S} \to \mathbb{B}$ as $\mathfrak{B}(\mathbf{x}) = \mathbf{x}/||\mathbf{x}'||$ for $\mathbf{x} \neq \mathbf{0}$, and $\mathfrak{B}(\mathbf{0}) = \mathbf{0}$. The mapping $\mathfrak{B}$ is illustrated in Figure 7.15.

Assuming nondegenerate simplices, $\mathfrak{B}$ is continuous and invertible, with $\mathbf{0}$ as its fixed point. Then, $\mathfrak{B}$ can be used to map $\vec{V}$ from any simplex in $\mathcal{M}$ onto an enclosing ball $\mathbb{B}$ with the center in the interior. We now show that if $\mathbf{0}$ is contained in the convex hull of $\vec{V}$, the Brouwer degree of this simplex with respect to $\mathbf{0}$ is odd.

Lemma 7.1. Let $\mathcal{S} = \{\mathbf{x}_0, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^n$ be a simplex of $\mathcal{M}$ containing the origin $\mathbf{0}$, and $\vec{V}$ be a vector field as defined above. If $\mathbf{0}$ lies in the convex hull of $\{\vec{V}_0, \ldots, \vec{V}_n\}$, then its Brouwer degree, $\deg(\vec{V}, \mathcal{S}, \mathbf{0})$, is odd.

Proof. Let $\mathbb{B}$ be the unit ball of $\mathcal{S}$ as defined above. Define a vector field $\vec{V}_{\mathbb{B}} : \mathbb{B} \to \mathbb{R}^n$, as $\vec{V}_{\mathbb{B}}(\mathbf{x}) = \vec{V}(\mathfrak{B}^{-1}(\mathbf{x}))$. Assume there exists an $\mathbf{x}_b \in \partial\mathbb{B}$ such that

$$\vec{V}_{\mathbb{B}}(\mathbf{x}_b) = a\vec{V}_{\mathbb{B}}(-\mathbf{x}_b), \text{ with } a > 0.$$

The following argumentation proves that this assumption leads to a contradiction.

Since $\mathfrak{B}$ is continuous, it cannot map adjacent points of $\mathcal{S}$ to antipodal points of $\mathbb{B}$, and, therefore, it follows that there exist two different facets of $\mathcal{S}$, namely $\mathcal{S}_j$ and $\mathcal{S}_k$, containing parallel vectors. Let $\mathbf{x} \in \mathcal{S}_j$ and $\mathbf{y} \in \mathcal{S}_k$, such that $\sum_{i \neq j} \alpha_i(\mathbf{x})\vec{V}_i = a \sum_{i \neq k} \alpha_i(\mathbf{y})\vec{V}_i$, $j \neq k$, for some $a > 0$.

$$\sum_{i \neq j} \alpha_i(\mathbf{x})\vec{V}_i - a \sum_{i \neq k} \alpha_i(\mathbf{y})\vec{V}_i = 0,$$

$$\sum_{i \neq j,k} \left(\alpha_i(\mathbf{x}) - a\alpha_i(\mathbf{y})\right) \vec{V}_i - a\alpha_j(\mathbf{y})\vec{V}_j + \alpha_k(\mathbf{x})\vec{V}_k = 0.$$

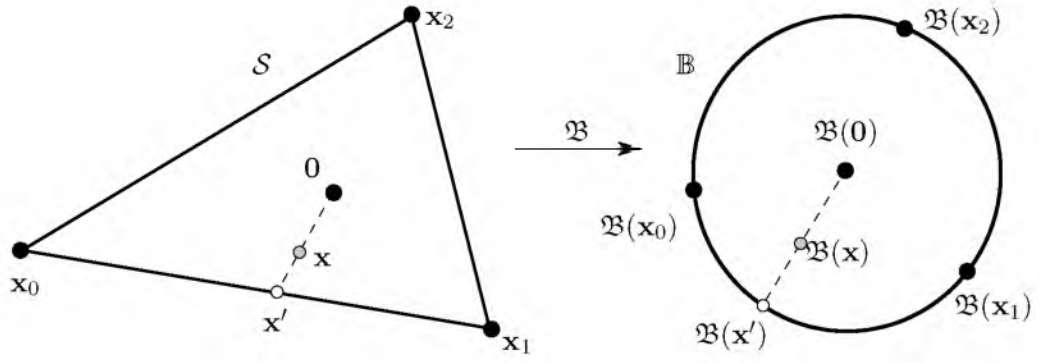


Figure 7.15. Mapping of a simplex to a unit ball to understand robust critical point detection, $\mathfrak{B} : \mathcal{S} \to \mathbb{B}$ in $\mathbb{R}^2$. The interior of $\mathcal{S}$ contains the origin $\mathbf{0}$.

Also, since $\mathbf{0}$ lies in the convex hull of $\{\vec{V}_0, \ldots, \vec{V}_n\}$, there exist weights c_i such that $\sum_{i=0}^{n} c_i \vec{V}_i = \mathbf{0}$, and $c_i \geq 0,\ \forall i$, and $\sum_{i=0}^{n} c_i = 1$. Now, since $\{\vec{V}_0, \ldots, \vec{V}_n\}$ forms an affine combination, $\sum \lambda_i \vec{V}_i = \sum \mu_i \vec{V}_i \Leftrightarrow \lambda_i = \mu_i\ \forall i$, which implies that

$$\begin{aligned} c_i &= \alpha_i(\mathbf{x}) - a\alpha_i(\mathbf{y}), && \forall i \neq j, k, \\ c_k &= \alpha_k(\mathbf{x}), \\ c_j &= -a\alpha_j(\mathbf{y}). \end{aligned}$$

However, since all c_i's, α_i's, and a are positive, the third condition gives a contradiction. Hence, a point $\mathbf{x}_b$ with $\vec{V}_{\mathbb{B}}(\mathbf{x}_b) = a\vec{V}_{\mathbb{B}}(-\mathbf{x}_b)$ does not exist. By Theorem 2.2, $\deg(\vec{V}_{\mathbb{B}}, \mathbb{B}, \mathbf{0})$, and hence $\deg(\vec{V}, \mathcal{S}, \mathbf{0})$ is odd. □

The above result allows proving the main result, which follows.

Theorem 7.1. Let $\mathcal{S} = \{\mathbf{x}_0, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^n$ be a simplex of $\mathcal{M}$ and $\vec{V}$ be a vector field on $\mathcal{M}$ as defined above. Then $\mathcal{S}$ contains a critical point if and only if $\mathbf{0}$ is in the interior of the convex hull of $\{\vec{V}_0, \ldots, \vec{V}_n\}$.

Proof. If $\mathbf{0}$ is in the interior of the convex hull of $\{\vec{V}_0, \ldots, \vec{V}_n\}$, then by Lemma 7.1, the Brouwer degree of the origin inside $\mathcal{S}$ is odd. Thus, by Theorem 2.1, there exists a critical point in $\mathcal{S}$.

If a critical point is located at $\mathbf{x}$ in $\mathcal{S}$, then

$$\begin{aligned} \vec{V}(\mathbf{x}) &= \sum_{i=0}^{n} \alpha_i(\mathbf{x}) \vec{V}_i = \mathbf{0}, \\ \Rightarrow \sum_{i=0}^{n} \frac{\alpha_i(\mathbf{x})}{\sum_{j=0}^{n} \alpha_j(\mathbf{x})} \vec{V}_i &= \sum_{i=0}^{n} c_i(\mathbf{x}) \vec{V}_i = \mathbf{0}. \end{aligned}$$

where $c_i(\mathbf{x}) = \frac{\alpha_i(\mathbf{x})}{\sum_{j=0}^{n} \alpha_j(\mathbf{x})}$. Since all α_i's are non-negative, it implies that $0 \leq c_i \leq 1,\ \forall i$, and $\sum_{i=0}^{n} c_i = 1$. Therefore, $\mathbf{0}$ is in the interior of the convex hull of $\vec{V}_i$'s. □

The practical implication of this result is that, to identify a simplex containing a critical point, one does not need to know the actual interpolation function as long as it satisfies the properties expressed by Equation (7.2). Furthermore, it suffices to check the convex hull of the vectors at the vertices to find singularities. Thus, detection of singularities for any interpolation scheme is reduced to a simpler PL test that can be performed using a combinatorial approach discussed in Section 7.2.2.

If the space is sampled too sparsely, or a more complex interpolation scheme is used, multiple critical points may appear within a single simplex. In this case, the presented

technique will assume the simplest possible interpretation of the vectors at the vertices. In particular, if the singularities in the same simplex cancel each other, then no singularities will be reported, since the vector field on the boundary can be completed with an entirely regular vector field in its interior.

Next, the topological consistency of the presented approach will be proved.

Theorem 7.2. Using Theorem 7.1 always leads to topologically consistent critical point detection for $\vec{V}$ defined on $\mathcal{M}$ as above.

Proof. Any smooth vector field on a closed orientable n-dimensional manifold $\mathbb{M}$ must have an even number of critical points. Therefore, given a simplicial complex $\mathcal{M}$ corresponding to $\mathbb{M}$, and a sampled vector field $\vec{V}$ defined on $\mathcal{M}$, consistency will be demonstrated by showing that the presented technique detects an even number of critical points for $\vec{V}$.

Consider $\mathcal{M}$ in function (vector) space, that is, consider the simplicial complex defined by the vectors of $\vec{V}$, and denote it as $\mathcal{M}_{\vec{V}}$. Note that an orientable and closed $\mathcal{M}$ implies an orientable and closed $\mathcal{M}_{\vec{V}}$. There exists an embedding of $\mathcal{M}_{\vec{V}}$ in $\mathbb{R}^{n+1}$, where the $n+1^{\text{th}}$ component is chosen at random. Trace a line ℓ through the origin and orthogonal to the $\mathbb{R}^n$ subspace. The line ℓ corresponds to zero vector $\mathbf{0}$. By Theorem 7.1, singularities are not allowed to exist on the faces of simplices of $\mathcal{M}$. Since $\mathcal{M}_{\vec{V}}$ is closed, there exist an even number of transveral intersections between $\mathcal{M}_{\vec{V}}$ and ℓ. Consequently, each intersection corresponds to a single simplex containing a critical point at its interior, leading to an even number of simplices with critical points, and, therefore, an even number of critical points. □

7.2.2 Robust Computation

Detection of critical point in a simplex $\mathcal{S}$ requires a robust way of determining whether the zero vector is contained in the convex combination of the vectors of $\mathcal{S}$. Let $\mathcal{S}_{\vec{V}}$ denote the simplex created by the vectors of $\vec{V}$ at the vertices of $\mathcal{S}$. Then, the detection of critical point simply translates into testing whether the origin $\mathbf{0}$ lies inside $\mathcal{S}_{\vec{V}}$. This point-in-simplex test can be further reduced to computing the orientation of the test point with respect to the facets of the simplex. This section describes how this procedure can be performed in a combinatorial manner.

7.2.2.1 Robust computation of orientation. The orientation of $n+1$ points in $\mathbb{R}^n$ can be computed as the sign of the following determinant.

$$\det(\mathbf{x}_0, \ldots, \mathbf{x}_n) = \begin{pmatrix} \mathbf{x}_{0,0} & \ldots & \mathbf{x}_{0,n-1} & 1 \\ \vdots & \ddots & \vdots & 1 \\ \mathbf{x}_{n,0} & \ldots & \mathbf{x}_{n,n-1} & 1 \end{pmatrix}.$$

The determinant is zero if all of the points lie on a common hyperplane. For example, in 2D, the orientation of three points is counterclockwise (positive) if the sign of the corresponding determinant is positive, or clockwise (negative) if the sign is negative. The degenerate case where the three points are collinear leads to the determinant being zero. Recall that the sign of a determinant switches if an odd number of row-exchanges are carried out; therefore, care must be taken with respect to the order of the points. Algorithm 7.1 [49] calculates the orientation of a set of points, by first assigning them a consistent order (that is, sorting), and then computing the sign of the determinant.

Algorithm 7.1 Positive$_n(\mathbf{x}_{j_0}, \ldots, \mathbf{x}_{j_n})$.

$s \leftarrow$ Sort $(j_0, \ldots, j_n)$: s is the number of swaps needed to sort
$d \leftarrow$ sign $\det(\mathbf{x}_{j_0}, \ldots, \mathbf{x}_{j_n})$
if odd(s) **then**
  $d \leftarrow -d$
end if
return d

It is easy to confirm that this determinant is always nonzero for generic vectors. To impose genericity, the Simulation of Simplicity (SoS) [49] is used. The SoS is a general-purpose programming technique to handle degeneracies in the data. It applies a symbolic perturbation to the data, preventing any of the determinants from becoming zero, and, thus, providing a nondegenerate point-in-simplex test. Intuitively, if a point lies on a shared coface of two or more simplices, the SoS makes a consistent choice by forcing it to be contained inside one of them, and outside the rest of the simplices.

Numerical robustness is achieved by first converting the data to a fixed-precision format, and then representing the values as long integers in computing the determinant (as done by the Geomdir library [146]), hence removing the need for floating-point arithmetic. As a result, the determinant computation is both robust and combinatorial. It should be noticed that Algorithm 7.1 is the fundamental step in the process of critical point detection, and helps achieve robustness by replacing numerical data (vector components) with combinatorial information (orientation).

7.2.2.2 Robust point-in-simplex test. Using Algorithm 7.1 for robust computation of orientation, Algorithm 7.2 [49] performs the point-in-simplex test. Given a simplex

$\mathcal{S} = \{\mathbf{x}_0, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^n$, the first step is to determine the orientation of the facets of the simplex. Then, one by one each facet is replaced by the given test point and the corresponding orientation is computed. If all such combinations have the same orientation, then the test point lies inside the simplex, otherwise not.

7.2.3 Experimental Results

This section presents the results of the presented technique to identify critical points in PL vector fields on a triangulated domain. For each simplex in the domain, whether the origin is contained inside the simplex created by the vectors at the vertices of the domain is determined. The Geomdir library [146] that uses the SoS was used to compute the determinants robustly. The obtained results are compared with the numerical method to detect critical points [118] that solves a linear system for every simplex in the domain.

To evaluate the two techniques, a synthetic 2D vector field with known critical points is created. Two of the critical points are placed on an edge and a vertex of the triangulation. Figure 7.16 shows that for both these critical points, the numerical technique determined that all triangles sharing the corresponding edge and vertex, respectively, contain critical points—a topologically inconsistent result. However, the proposed method uses the SoS to make a choice for using a single triangle for representing each critical point.

To demonstrate dimension-independence of the presented technique, it is tested on the Lorenz system (A 3D vector field $\vec{V}(x, y, z) = \{\sigma(y - x),\ x(\rho - z) - y,\ xy - \beta z\}$) with parameters $\sigma = 10$, $\beta = 8/3$, and $\rho = 1/2$. For $\rho < 1$, the system contains a singularity at the origin. Figure 7.17 shows a comparison where the presented technique selects a single tetrahedron touching the origin to represent the singularity. The numerical technique, on the other hand, detects all 20 tetrahedra touching the origins as critical.

To compare the running times of the two methods, the flow representing the 2D global

Algorithm 7.2 Point $\mathbf{x}$ in Simplex $\mathcal{S} = \{\mathbf{x}_0, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^n$.

```
s ← Positive_n(x_0, ..., x_n)
for i = 0 to n do
  (x'_0, ..., x'_n) ← (x_0, ..., x_n)
  x'_i ← x
  s_i ← Positive_n(x'_0, ..., x'_n)
  if s ≠ s_i then
    return false
  end if
end for
return true
```

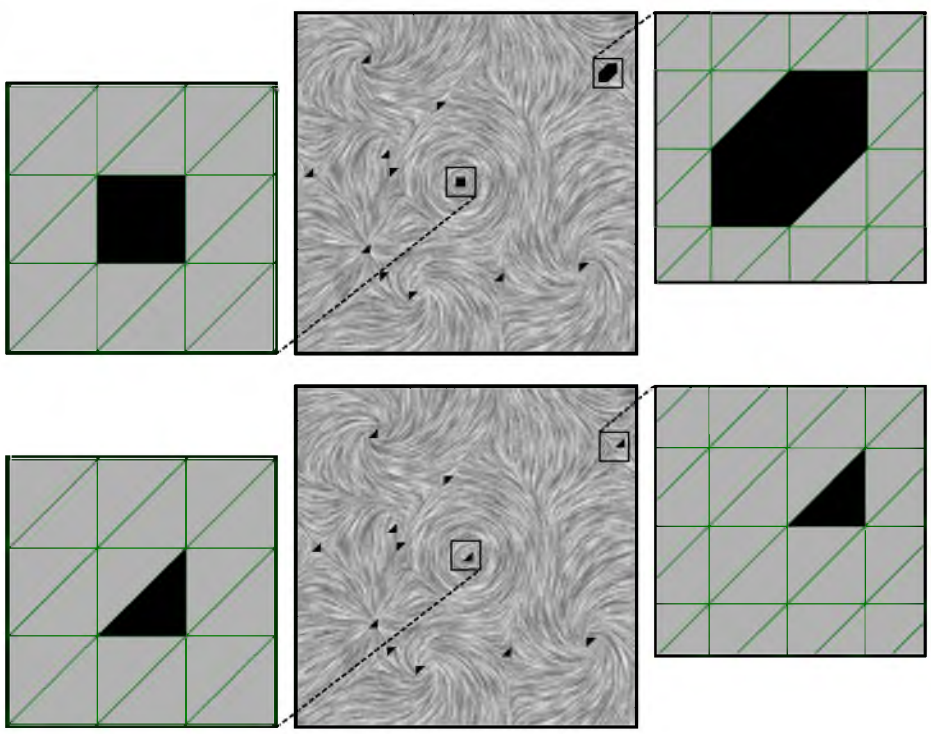


Figure 7.16. Consistent detection of singularities in a 2D analytic vector field. The top row shows the triangles where critical points were detected numerically. Zoom-in views show that multiple triangles test positive for two critical points lying on the boundaries of triangles. The bottom row shows the consistent results detected using our algorithm where only one triangle per critical point tested positive.

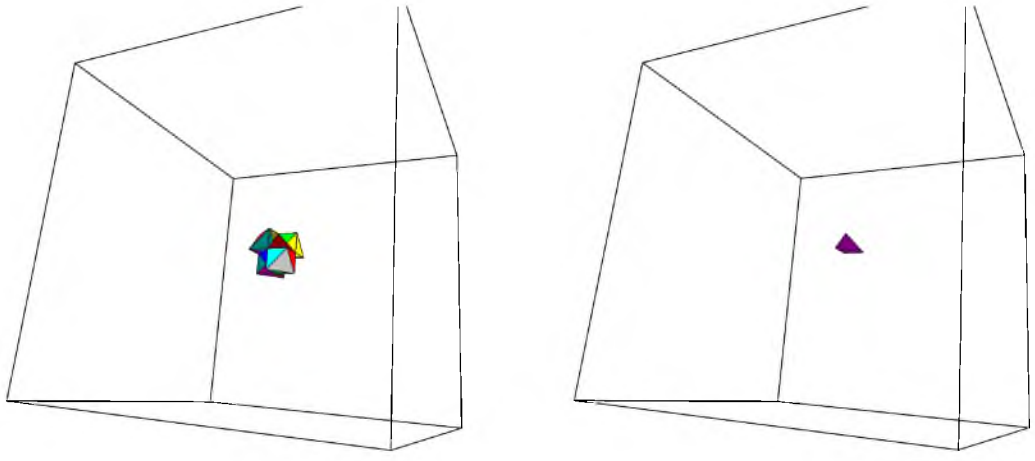


Figure 7.17. Consistent detection of singularities in a 3D Lorenz vector field. Numerical method (left) detects 20 tetrahedra touching the origin, whereas our technique (right) detects a single tetrahedron to represent the singularity.

oceanic currents (see Data B.3.3) was used. The ocean surface is represented by $10,633,760$ triangles. The numerical method [118] takes ≈ 89.22 seconds to detect $24,552$ critical points, whereas the presented method takes only ≈ 6.85 seconds to detect the same critical points (as illustrated in Figure 7.18), and, thus, is significantly faster.

7.3 Summary

This chapter presents two techniques for consistent analysis of vector fields—quantized Edge Maps and combinatorial critical point detection. Both approaches are based on discrete theories, and are combinatorial in nature, so robust algorithms can be applied for feature detection. Combining these techniques, consistent topology is demonstrated next in Figures 7.18 to 7.25. The first flow used is the top slice of the flow representing 3D global oceanic currents (see Data B.3.3) followed by the 2D ocean winds flow (see Data B.1.2) and the 2D flow inside the HCCI engine (see Data B.1.1).

The quantized Edge Maps represent the flow by discretizing the flow along the edges of a triangulation, and constructing Edge Maps on top of this representation. As a result, all the advantages of the Edge Maps are naturally carried over to the new quantized representation. This quantized flow represents an important step towards a new theory of vector fields by combining discrete algorithms with the approximation power of interpolated fields. This representation enables graph-based algorithms that extract typical flow structures consistently and with geometric fidelity. Consequently, they have the potential to form the basis of a new set of analysis techniques that repeat the recent success of discrete scalar field topology.

The current chapter focuses on PL vector fields and triangulated surfaces; however, one of the advantages of this new representation is that it is easily extensible to different mesh types and interpolation schemes. By reserving more bits to encode the edge number on an interval, the existing datastructure can be used to encode flow on any polygonal mesh. Furthermore, the flow is queried in just two clearly defined ways: to find zero-crossings and to find exit points of streamlines. Thus, the input flow representation could be a black box solver, making it amenable to, for example, higher-order interpolation schemes. The main adjustment necessary would be a more general conversion algorithm as some of the convenient assumptions, for example, the limited number of transition points per edge, would no longer hold. In this manner, quantized flow forms a unifying representation able to separate the flow encoding used by a simulation from the analysis within a strict and well-known approximation error. A complete understanding of the theoretical nature of

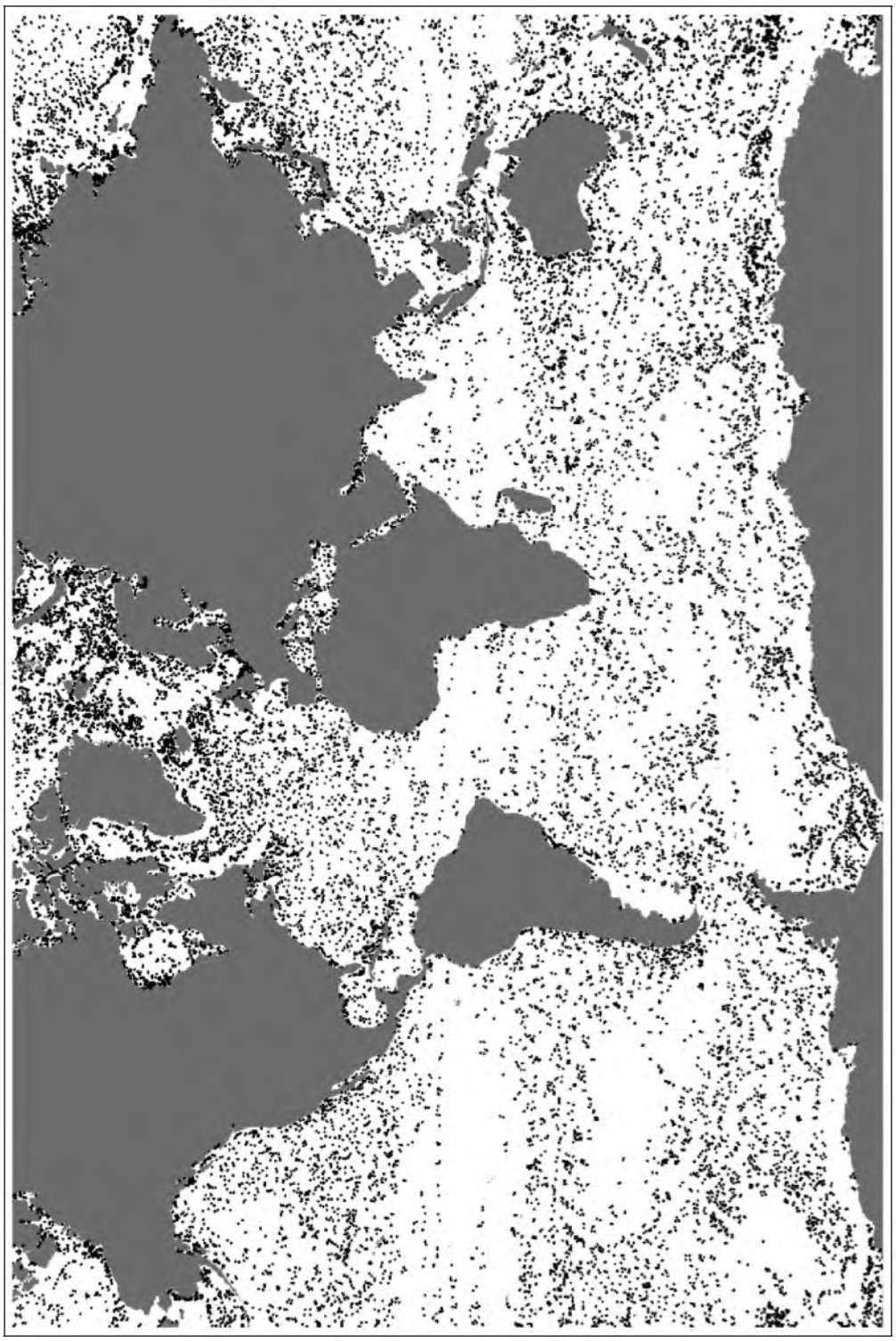


Figure 7.18. Consistent critical points of the 2D global oceanic currents detected 24,552 triangles with critical points, each represented by a black dot.

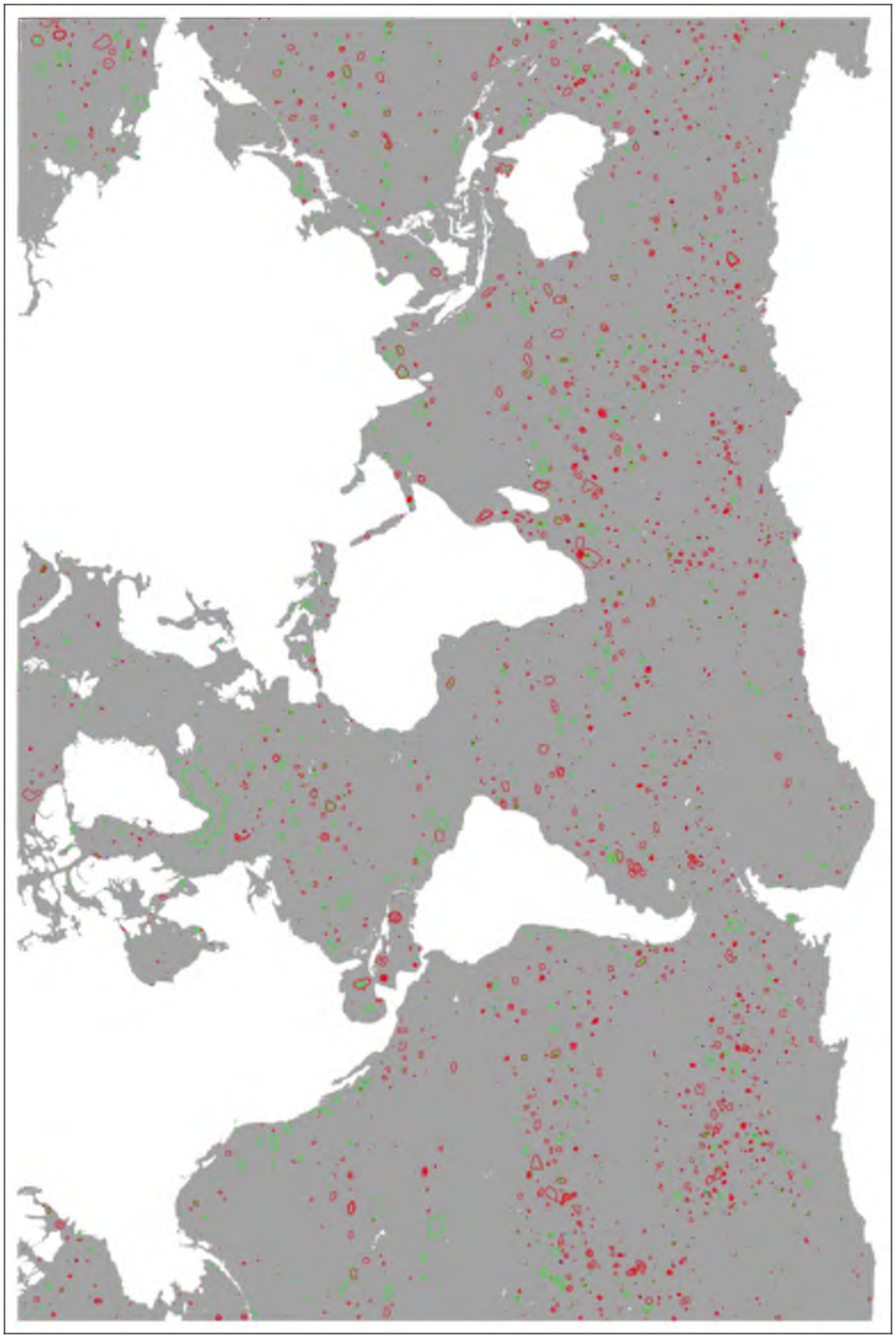


Figure 7.19. Consistent periodic orbits of the 2D global oceanic currents detected $\approx 39,000$ orbits.

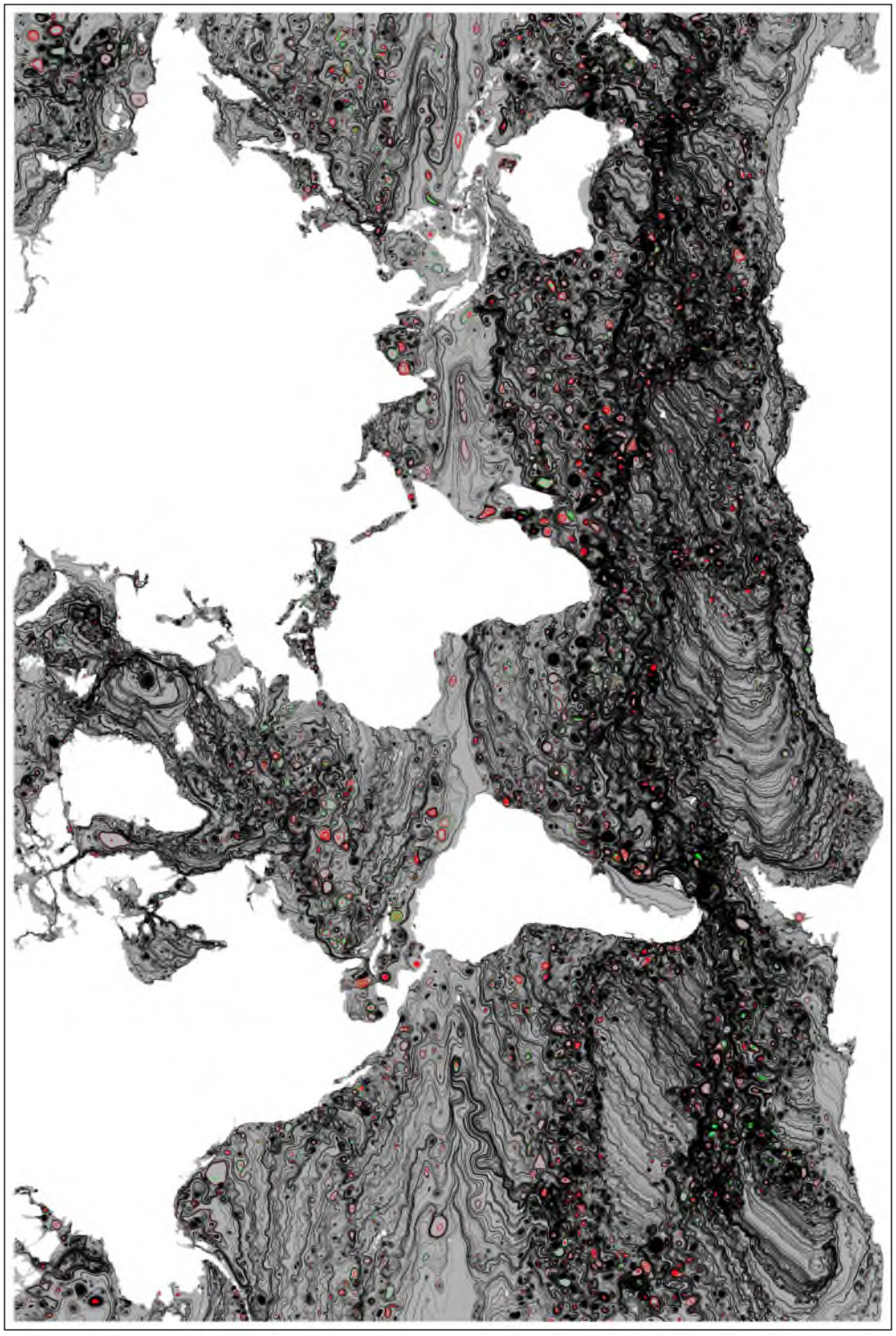


Figure 7.20. Consistent topological skeleton of the 2D slice of the global oceanic currents data.

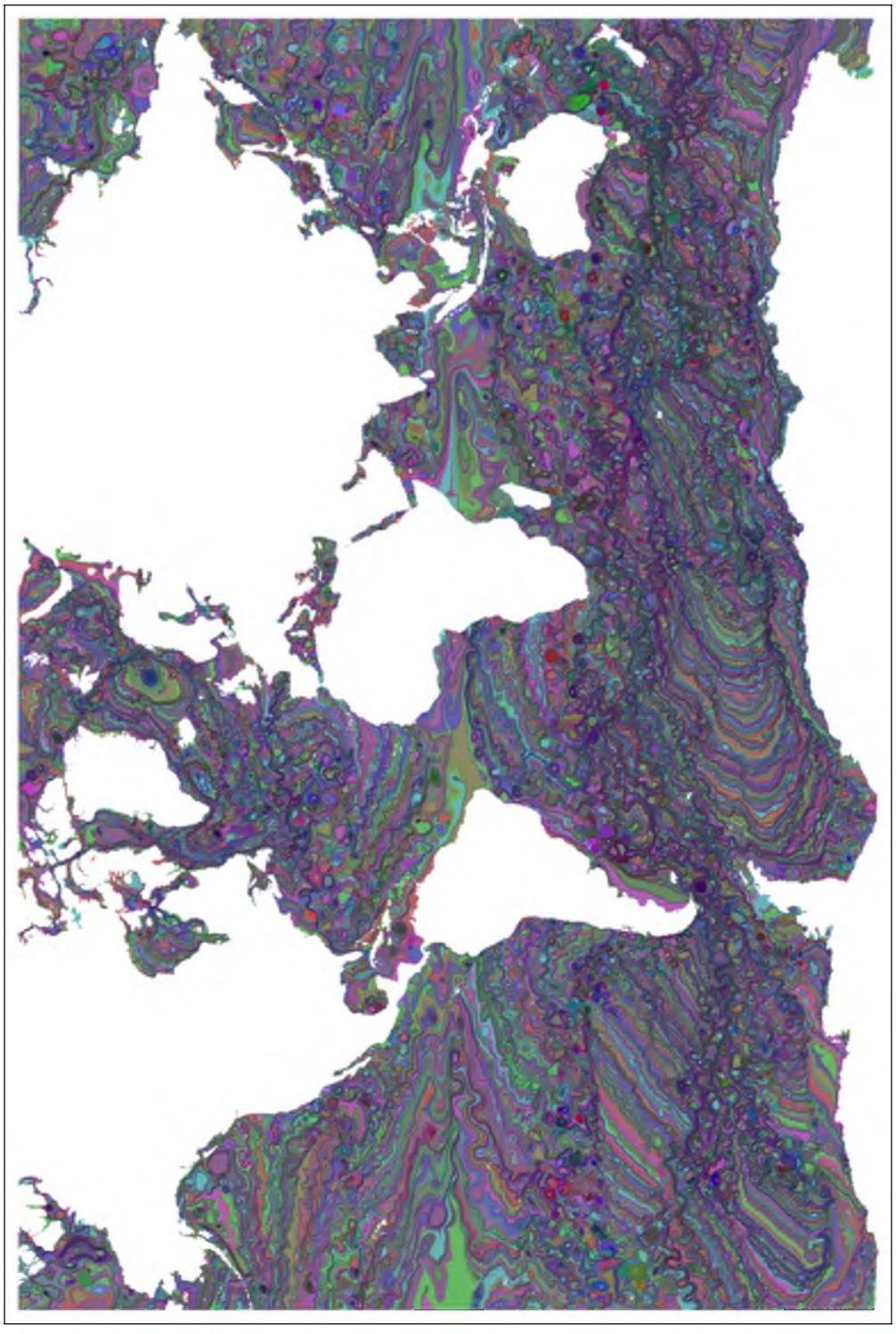


Figure 7.21. Consistent (unstable) manifolds of the 2D global oceanic currents data, each represented by a different color.

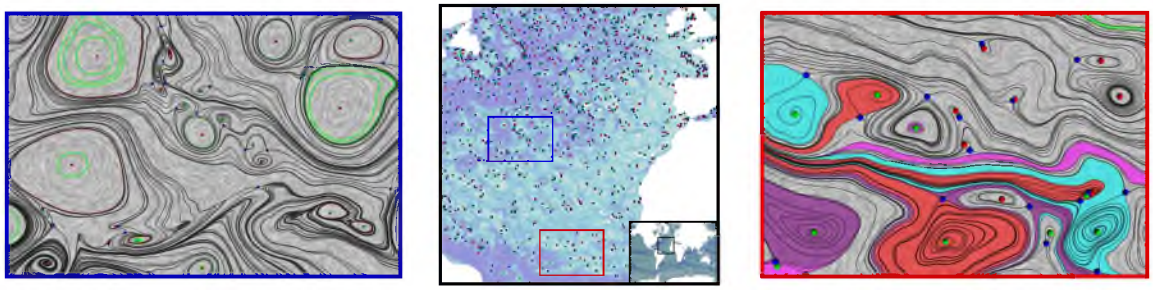


Figure 7.22. Consistent topology in the North Atlantic region of the 2D global oceanic currents. The region is discretized as a $[600 \times 600]$ regular grid. (Left) Black lines are separatrices grown from all saddles (blue balls), and green curves show detected closed orbits. (Right) Different colored regions are unstable manifolds grown from all the sources (green balls).

these approximation errors remains future work.

The second part of this chapter provides a necessary and sufficient condition for the existence of critical points in a simplex for a broad class of interpolated vector fields. This existence condition for critical points allows developing a robust method to detect critical points in nD. Furthermore, when given finite-precision values, the technique is guaranteed to use finite precision, and, therefore, the result can be computed exactly in a combinatorial manner. In the future, it will be useful to investigate further the class of interpolation functions for which these results were shown, identifying which other interpolation techniques fall into this class. It will also be useful to classify critical points in a similar combinatorial approach.

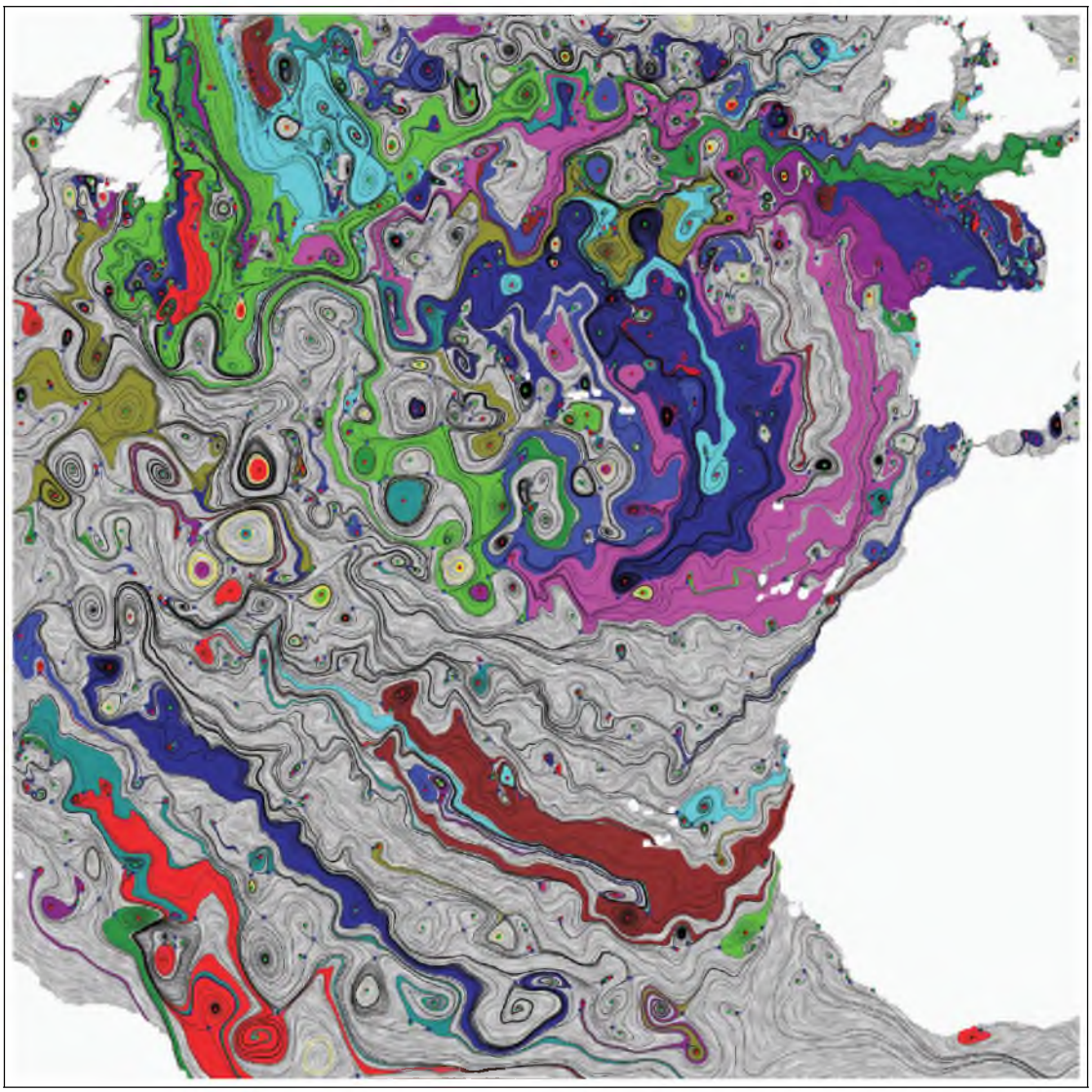


Figure 7.23. Consistent unstable manifolds in the North Atlantic region of the 2D global oceanic currents. These structures are different from those in Figure 7.22 (right) due to different error thresholds of the underlying maps. Nevertheless, irrespective of the accuracy, the results are consistent.

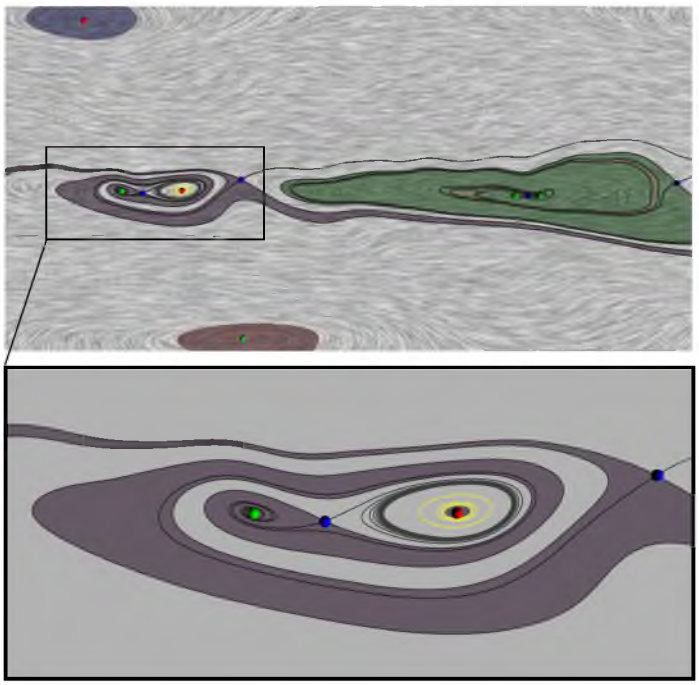


Figure 7.24. Consistent topology of the 2D ocean winds flow (see Data B.1.2). Orbits and separatrices are shown in yellow and black, respectively.

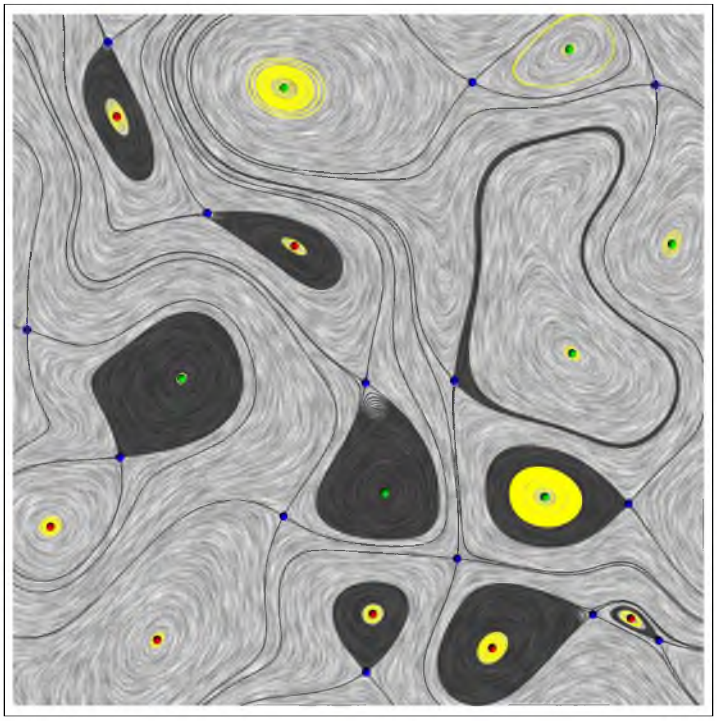


Figure 7.25. Consistent topology of the 2D flow inside HCCI engine (see Data B.1.1). Orbits and separatrices are shown in yellow and black, respectively.

PART IV

CONCLUSION

CHAPTER 8

SUMMARY AND OUTLOOK

This dissertation presents new theories and frameworks to perform consistent feature extraction from vector fields. These techniques enable analysis applications to answer two important concerns during feature extraction: (1) *"Are the extracted features correct?"* and (2) *"Is the extraction procedure precise?"*. As a result, the work presented here is directly useful for gaining scientific understanding of a variety of physical phenomena that can be represented using vector fields, for example ocean currents [133], wind patterns [15], turbulent combustion [73, 222], solar convection in astrophysics [194], etc.

One of the most fundamental challenges in vector field analysis is its dependence upon reference frames. Most features of vector fields are defined with respect to the assumed reference frame. As a result, the definition of "correct" features is subjective. Part II of this work promotes the idea that each analysis task must define its own preferred reference frame, in which the extracted features are most useful. This work presents new reference frames that focus on the local properties of the flow.

Chapter 3 develops a new formulation of a known vector field decomposition. This so-called natural Helmholtz-Hodge decomposition (HHD) decomposes a given vector field into three components, uniquely and without imposing any boundary conditions. As a result, any analysis using the natural HHD mimics an open boundary useful in many applications. Most of the existing applications that compute the conventional formulations of the HHD make use of the two nonharmonic components of the decomposition, and typically discard the harmonic component as error. Chapter 4, for the first time, describes the utility of the harmonic component obtained through the natural HHD to represent a local reference frame, which allows extracting important features that cannot be detected using other techniques and in other frames.

These local reference frames have an important additional advantage. Most vector fields of practical interest are unsteady (time-varying) in nature, and their analysis poses serious challenges—both fundamental and computational. Transforming a given unsteady vector

field into this local reference frame simplifies its analysis by treating it as a sequence of steady (time-independent) vector fields. Features of unsteady vector fields are spatiotemporal in nature, and as such, spatial analysis is incapable of detecting these features. However, enabling the representation of unsteady fields as sequences of steady fields decomposes their spatiotemporal analysis into two separate components: spatial and temporal. As a result, such flows can be analyzed through a spatial (streamline-based) analysis, followed by a temporal collation of the spatial features. Chapter 5 gives a case study to motivate such analysis by demonstrating that commonly accepted ideas of using pathlines for understanding unsteady flows are unreliable.

Part III of this dissertation addresses the second question asked above, and focuses on the numerical robustness of analysis techniques. Commonly used techniques for vector field analysis are susceptible to numerical errors, leading to the problems of inaccuracy and inconsistency. This dissertation argues that consistency is the *sine qua non* of any computational algorithm, since lack of consistency with the underlying theory can have far graver consequences than inaccuracy.

Chapters 6 and 7 develop a new discrete theory of vector fields. This theory is implemented using Edge Maps, which are a new representation that allows computing the streamlines of a steady vector field in a guaranteed consistent manner. The fundamental philosophy behind obtaining consistency is to impart memory to the application, so that every analysis decision is consistent with all the decisions made in the past. The Edge Maps implement this idea by dividing triangles in the given mesh into smaller pieces that represent mutually nonintersecting flow—an initial decision made by analysis. Subsequently, a streamline passing through one of the pieces can be guaranteed to stay inside, that is, not intersect with any other piece. The number of pieces in this triangle decomposition dictates the available geometric resolution; nevertheless, the flow is always consistent irrespective of the amount of subdivision. Thus, Edge Maps represent the flow through a single triangle in a consistent manner. By induction over triangles, streamline tracing, which is the most fundamental task in streamline-based analysis of steady vector fields, can be performed in a consistent manner. Chapter 7 also presents a combinatorial framework for consistent detection of critical points. Based on a similar philosophy, the framework uses symbolic perturbations to "remember" the triangles that contain critical points, making possible their consistent detection. Together with consistent streamline tracing, this framework allows consistent feature extraction in streamline-based analysis.

8.1 Scope of This Dissertation

As discussed in Chapter 1, vector fields are versatile in nature, in that they can represent many types of data and phenomena. Although most of the results of the techniques presented in this dissertation have been demonstrated on vector fields that represent flow of material, these techniques are more generally applicable to other types of vector fields.

In particular, the ideas of local reference frames are directly applicable to other types of vector fields. For example, one can imagine extracting the behavior of a local force field (gravitational, magnetic, etc.) in the presence of a more globally existing force field. Similarly, understanding dynamical systems may require studying local influences, in which case the proposed reference frames may be used. These techniques may also be applied to isotropic vector fields found in electromagnetics, or cosmological vector fields. Furthermore, the discrete representation is also applicable and useful for different types of vector fields where one may require streamline tracing—almost all vector fields.

Currently, this work is limited in the spatial dimensionality of vector fields. The work presented in Part II is built upon the fundamental ideas of the Helmholtz-Hodge decomposition, which is defined for vector fields in 2D and 3D; as a result, our formulations and definitions are also limited to 2D and 3D. However, there may be ways to generalize the techniques presented in this dissertation to higher dimensions using more general vector field decompositions [86, 87, 109, 190]. These ideas will require further investigation.

8.2 Directions for Future Research

Although the work described in this dissertation, in its current form, is useful for a variety of applications, there are a number of ways to improve it and make it even more widely applicable. First of all, this dissertation proposes the idea of boundary-to-boundary flow mapping, and applies it only to 2D triangular meshes (in Chapters 6 and 7). Nevertheless, as a concept, the same idea can be applied to other forms of discretizations and/or interpolation schemes. In particular, developing Edge Maps for quadilateral cells with bilinearly interpolated flow will be especially interesting, since many simulations' output data defined on regular grids. With a change in interpolation, however, the properties and construction of Edge Maps will be different, since a bilinear interpolation does not fulfill the properties of piecewise linear flow. In particular, simplifying assumptions such as a single transition point per-edge will not be true anymore, and a fresh study of possible equivalence classes must be performed.

Another research direction is to study the Edge Maps in 3D, in which case they should instead be called "Face Maps". The construction of 3D maps appears to be particularly

challenging since one may be required to represent arbitrarily shaped subsets of faces of tetrahedra and map them to other similar pieces of possibly different shapes. However, constructing the Face Maps may be possible through appropriate "gridding" (similar to binning presented in Chapter 7) of the faces, and snapping the flow to the nearest grid point. Then, the next step would be to understand how to encode them concisely, since moving from 2D to 3D will increase the memory footprint of this representation substantially. Yet another interesting research direction is to develop similar representations for unsteady vector fields. For the 2D case, one may be able to use the Edge Maps framework for steady vector fields that is presented in this dissertation. Nevertheless, since now the flow changes over time, every triangle will require multiple Edge Maps—one for each time-step. The most important challenge then becomes interpolating between these Edge Maps to trace pathlines. When these ideas are implemented, however, they will provide a first-of-its-kind consistent framework for pathline tracing and the associated topological structures in unsteady vector fields. These directions require deeper thinking and substantial research efforts—an excellent opportunity for a new graduate student for a successful and impactful Ph.D. dissertation.

Chapter 4 of this dissertation develops a new theory of internal reference frames for unsteady vector fields. Although the internal frames are already useful for many complex simulated flows, this work is only the tip of the iceberg in this new direction of research. In particular, the motion of the internal frames is represented as a harmonic flow, which, although more general than currently used uniformly moving frames, is still far away from addressing even more-general types of vector fields. A number of physical scenarios may contain different reference frames, which currently remain unknown. For example, consider a small vortex rotating inside a bigger vortex. None of the existing techniques (including the work presented in this dissertation) may be able to differentiate between and detect both the vortices, even at different scales. There is a need to develop techniques that can automatically provide a multiscale analysis of unsteady flows, and in the process, determine the "correct" reference frame for every scale. Defining the "correct" reference frame in this context is also still unclear, and provides an excellent direction for future research. It appears that a more-general reference frame, such as an expanding or a rotating frame, will address more-general types of flows. In particular, the famous examples of the double gyre [187] and the petri dish [218] may provide good litmus tests for more-general reference frames.

8.3 Future Impact

The work presented in this dissertation is a fundamentally new way of representing and analyzing vector fields. Not only does it improve the state of the art in vector field analysis, this work also makes the existing well-established steady vector field analysis techniques accessible to unsteady vector fields—the more prevalent form of vector fields. Especially since the analysis of steady vector fields is more advanced and better understood than the analysis of unsteady ones, the ideas presented in this dissertation offer significant advances for the analysis of the latter. Using the ideas presented in this dissertation, unsteady vector fields can be analyzed on a per time-step basis, thus allowing temporal locality, and hence, better data management and higher levels of parallelization. These ideas also create a potential for in situ analysis of large-scale flow phenomena. Analyzing such flows is integral in a variety of applications across scientific domains. A number of contemporary challenges in science and engineering—for example, development of fuel-efficient designs of combustion engines, understanding of critical processes that drive climate changes, and exploring efficient renewable energy sources—can benefit from the techniques developed in this work.

APPENDIX A

DERIVATIONS

A.1 The Helmholtz-Hodge Decomposition (HHD)

A.1.1 Proof of Existence of HHD

- For the Poisson system (2.20).

$$\Delta D = \nabla \cdot \vec{V} \quad \text{on } \Omega$$
$$\vec{n} \cdot \nabla D = \vec{n} \cdot \vec{V} \quad \text{on } \partial\Omega$$

In this case, the divergence theorem [71],

$$\int_{\Omega} \nabla \cdot \vec{V} \ \mathrm{d}\mathbf{x} = \int_{\partial\Omega} \vec{n} \cdot \vec{V} \ \mathrm{d}\mathbf{x}$$

ensures that the compatibility condition for the Poisson equation is satisfied, and hence, it has a solution D, which is unique up to a constant leading to the existence of a unique $\vec{d}(= \nabla D)$ and thus a unique $\vec{r} = \vec{V} - \vec{d}$. The compatibility condition associated with the Neumann boundary condition is discussed in Section 2.1.

- For the Poisson system (2.19).

$$\vec{\nabla}^2 \vec{R} = -\nabla \times \vec{V} \quad \text{on } \Omega$$
$$\vec{n} \times \nabla \times \vec{R} = \vec{n} \times \vec{V} \quad \text{on } \partial\Omega$$

Once again, it can be shown that the Green's theorem [71] satisfies the compatibility condition. Thus, it admits a solution $\vec{R}$ which is unique up to a constant leading to the existence of a unique $\vec{r} = \nabla \times \vec{R}$, and, thus, a unique $\vec{d} = \vec{V} - \vec{r}$.

A.1.2 Orthogonality of the HHD

The orthogonality with respect to the L_2 inner product $\left(\langle f, g \rangle = \int f \cdot g\right)$ is established by showing that

$$\int_{\Omega} \vec{d} \cdot \vec{r} \ \mathrm{d}\mathbf{x} = 0. \tag{A.1}$$

Corresponding to the different boundary conditions, it is shown that (1) the space of gradient vector fields is orthogonal to the space of divergence-free vector fields that are parallel to

the boundary, and (2) the space of divergence-free vector fields is orthogonal to the space of gradient vector fields that are normal to the boundary.

- In the case of boundary conditions (2.17) ($\vec{n} \cdot \vec{r} = 0$),

$$\begin{aligned}
\int_\Omega \vec{d} \cdot \vec{r} \ \mathrm{d}\mathbf{x} &= \int_\Omega \nabla D \cdot \vec{r} \ \mathrm{d}\mathbf{x} \\
&= \int_\Omega \nabla\cdot(D\vec{r}) - D(\nabla\cdot\vec{r}) && \text{(by chain rule)} \\
&= \int_\Omega \nabla D \cdot \vec{r} && (\text{as } \nabla\cdot\vec{r} = 0) \\
&= \int_\Omega \nabla\cdot(D\vec{r}) \ \mathrm{d}\mathbf{x} \\
&= \int_{\partial\Omega} D \ \vec{r}\cdot\vec{n} \ \mathrm{d}\mathbf{x} && \text{(by divergence theorem)} \\
&= 0
\end{aligned}$$

- In the case of boundary conditions (2.18) ($\vec{n} \times \vec{d} = 0$),

$$\begin{aligned}
\int_\Omega \vec{d} \cdot \vec{r} \ \mathrm{d}\mathbf{x} &= \int_\Omega \vec{d} \cdot \left(\nabla \times \vec{R}\right) \ \mathrm{d}\mathbf{x} \\
&= \int_\Omega \nabla\cdot\left(\vec{R} \times \vec{d}\right) \ \mathrm{d}\mathbf{x} && (\vec{A}\cdot(\vec{B}\times\vec{C}) = (\vec{A}\times\vec{B})\cdot\vec{C}) \\
&= \int_{\partial\Omega} \left(\vec{R} \times \vec{d}\right)\cdot\vec{n} \ \mathrm{d}\mathbf{x} && \text{(by Gauss theorem)} \\
&= \int_{\partial\Omega} \vec{R}\cdot\left(\vec{d}\times\vec{n}\right) \ \mathrm{d}\mathbf{x} && (\vec{A}\cdot(\vec{B}\times\vec{C}) = (\vec{A}\times\vec{B})\cdot\vec{C}) \\
&= 0
\end{aligned}$$

Thus, for the decomposition to be L_2-orthogonal, the boundary conditions (2.17), (2.18), (2.19), and (2.20) are all sufficient, but not necessary. The orthogonality may be established with other types of boundary conditions too, as long as Equation (A.1) is satisfied.

A.1.3 Uniqueness of the HHD

As already discussed, the boundary conditions imposed on the decomposition determine the uniqueness. It is important to note that the two boundary conditions (2.17) (or (2.20)) and (2.18) (or (2.19)) may give different unique decompositions, as the harmonic component is represented differently. Here, we show the uniqueness of the decomposition corresponding to these boundary conditions.

- Assume that two different orthogonal decompositions of a vector field (with respect to boundary condition (2.17)) exist,

$$\begin{aligned} \vec{V} &= \nabla D_1 + \vec{r}_1 = \nabla D_2 + \vec{r}_2 \\ \Rightarrow 0 &= \vec{r}_1 - \vec{r}_2 + \nabla(D_1 - D_2) \end{aligned} \tag{A.2}$$

Taking the inner product of Equation (A.2) with $\vec{r}_1 - \vec{r}_2$ gives

$$\begin{aligned} 0 &= \int_\Omega \|\vec{r}_1 - \vec{r}_2\|^2 + (\vec{r}_1 - \vec{r}_2) \cdot \nabla(D_1 - D_2)\, \mathrm{d}\mathbf{x} \\ &= \int_\Omega \|\vec{r}_1 - \vec{r}_2\|^2\, \mathrm{d}\mathbf{x} + \int_\Omega (\vec{r}_1 - \vec{r}_2) \cdot \nabla(D_1 - D_2)\, \mathrm{d}\mathbf{x} \\ &= \int_\Omega \|\vec{r}_1 - \vec{r}_2\|^2\, \mathrm{d}\mathbf{x} + \int_\Omega (\vec{r}_1 \cdot \nabla D_1 + \vec{r}_2 \cdot \nabla D_2 - \vec{r}_1 \cdot \nabla D_2 - \vec{r}_2 \cdot \nabla D_1)\, \mathrm{d}\mathbf{x} \\ &= \int_\Omega \|\vec{r}_1 - \vec{r}_2\|^2\, \mathrm{d}\mathbf{x} + 0 - \int_\Omega (\vec{r}_1 \cdot \nabla D_2 + \vec{r}_2 \cdot \nabla D_1)\, \mathrm{d}\mathbf{x} && \text{(by orthogonality)} \\ &= \int_\Omega \|\vec{r}_1 - \vec{r}_2\|^2\, \mathrm{d}\mathbf{x} - \int_{\partial\Omega} (D_2\ \vec{r}_1 \cdot \vec{n} + D_1\ \vec{r}_2 \cdot \vec{n})\, \mathrm{d}\mathbf{x} && \text{(by divergence theorem)} \\ &= \int_\Omega \|\vec{r}_1 - \vec{r}_2\|^2\, \mathrm{d}\mathbf{x} && \text{(since } \vec{r} \cdot \vec{n} = 0) \end{aligned}$$

Thus, the above equality is satisfied only if $\int_\Omega \|\vec{r}_1 - \vec{r}_2\|^2\ \mathrm{d}\mathbf{x} = 0 \Rightarrow \vec{r}_1 = \vec{r}_2$, which implies $\nabla D_1 = \nabla D_2$, which violates the assumption. Thus, the decomposition given by the boundary condition (2.17) is unique.

- Similarly, by taking the inner product of Equation (A.2) with $\nabla(D_1 - D_2)$, it can be shown that the decomposition given by the boundary condition (2.18) is unique.

A.1.4 Derivation of the Two-Component HHD on Finite Domains

The Expressions (2.16) for the two-component HHD on finite domains can be derived as follows:

$$\begin{aligned} \vec{V}(\mathbf{x}) &= \int \vec{V}(\mathbf{x}_0)\ \delta(\mathbf{x} - \mathbf{x}_0)\ \mathrm{d}\mathbf{x}_0 && \text{(by definition of Delta function)} \\ &= \int \vec{V}(\mathbf{x}_0)\ \nabla^2 \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \mathrm{d}\mathbf{x}_0 && (\nabla^2 \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) = \delta(\mathbf{x} - \mathbf{x}_0)) \\ &= \int \nabla^2 \left(\mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0) \vec{V}(\mathbf{x}_0) \right)\ \mathrm{d}\mathbf{x}_0 && (\vec{V}(\mathbf{x}_0) \text{ is constant w.r.t. } \nabla) \\ &= \int \left(\nabla\nabla \cdot \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \vec{V}(\mathbf{x}_0)\ - \nabla \times \nabla \times \mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \vec{V}(\mathbf{x}_0) \right)\ \mathrm{d}\mathbf{x}_0 \\ & && (\nabla^2 = \nabla\nabla\cdot - \nabla\times\nabla\times) \\ &= \nabla \int \nabla \cdot \left(\mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \vec{V}(\mathbf{x}_0) \right)\ \mathrm{d}\mathbf{x}_0 - \nabla \times \int \nabla \times \left(\mathrm{G}_\infty(\mathbf{x}, \mathbf{x}_0)\ \vec{V}(\mathbf{x}_0) \right)\ \mathrm{d}\mathbf{x}_0 \\ &= \nabla D(\mathbf{x}) + \nabla \times \vec{R}(\mathbf{x}) \end{aligned}$$

In the following, $\vec{n}$ represents the exterior normal to the boundary. For simplicity in the following expressions, we swap $\mathbf{x}$ and ∇ with $\mathbf{x}_0$ and ∇_0, respectively. Consider the first integral term.

$$\begin{aligned}
D(\mathbf{x}_0) &= \int \nabla_0 \cdot \Big(\mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \vec{V}(\mathbf{x})\Big)\ \mathrm{d}\mathbf{x} && \text{(by definition of } D\text{)}\\
&= \int \vec{V}(\mathbf{x}) \cdot \nabla_0 \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \mathrm{d}\mathbf{x} && (\vec{V}(\mathbf{x}) \text{ is constant wrt } \nabla_0)\\
&= -\int \vec{V}(\mathbf{x}) \cdot \nabla \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \mathrm{d}\mathbf{x} && (\nabla \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) = -\nabla_0 \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0))\\
&= \int \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) \Big(\nabla \cdot \vec{V}(\mathbf{x})\Big)\,\mathrm{d}\mathbf{x} - \int \nabla \cdot \Big(\mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \vec{V}(\mathbf{x})\Big)\,\mathrm{d}\mathbf{x}\\
& && ((\nabla f)\cdot\vec{V} = \nabla\cdot(f\vec{V}) - f(\nabla\cdot\vec{V}))\\
&= \int \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) \Big(\nabla \cdot \vec{V}(\mathbf{x})\Big)\,\mathrm{d}\mathbf{x} - \oint \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \vec{n}\cdot\vec{V}(\mathbf{x})\ \mathrm{d}\mathbf{x}\\
& && \text{(by divergence theorem)}
\end{aligned}$$

Now, consider the second integral term.

$$\begin{aligned}
\vec{R}(\mathbf{x}_0) &= -\int \nabla_0 \times \Big(\mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \vec{V}(\mathbf{x})\Big)\ \mathrm{d}\mathbf{x} && \text{(by definition of } \vec{R}\text{)}\\
&= -\int \vec{V}(\mathbf{x}) \times \nabla_0 \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \mathrm{d}\mathbf{x} && (\vec{V}(\mathbf{x}) \text{ is constant wrt } \nabla_0)\\
&= \int \vec{V}(\mathbf{x}) \times \nabla \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \mathrm{d}\mathbf{x} && (\nabla \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) = -\nabla_0 \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0))\\
&= \int \nabla \times \Big(\mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0)\ \vec{V}(\mathbf{x})\Big)\,\mathrm{d}\mathbf{x} - \int \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) \Big(\nabla \times \vec{V}(\mathbf{x})\Big)\,\mathrm{d}\mathbf{x}\\
& && ((\vec{V}\times\nabla f) = \nabla\times(f\vec{V}) - f(\nabla\times\vec{V}))\\
&= \oint \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) \Big(\vec{n} \times \vec{V}(\mathbf{x})\Big)\,\mathrm{d}\mathbf{x} - \int \mathrm{G}_\infty(\mathbf{x},\mathbf{x}_0) \Big(\nabla \times \vec{V}(\mathbf{x})\Big)\ \mathrm{d}\mathbf{x}\\
& && \text{(by Gauss theorem)}
\end{aligned}$$

A.1.5 Simplified Expressions of the HHD in 2D

Equations (2.22) and (2.23) can be derived using the properties of $\mathcal{J}$. By definition, if $\vec{V} = (v_1, v_2)$, then $\mathcal{J}\vec{V} = (-v_2, v_1)$. Then, $\nabla\times\vec{V} = -\nabla\cdot\mathcal{J}\vec{V}$ as derived below.

$$\begin{aligned}
\nabla\cdot\vec{V} &= \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}\\
\nabla\times\vec{V} &= \frac{\partial v_2}{\partial x} - \frac{\partial v_1}{\partial y}\\
&= \nabla\cdot(v_2, -v_1)\\
&= \nabla\cdot(-\mathcal{J}(v_1, v_2))\\
&= -\nabla\cdot\mathcal{J}\vec{V}
\end{aligned}$$

Denoting a scalar field as S, its gradient ∇S is an irrotational vector field. Then, $\mathcal{J}\nabla S$ is an incompressible vector field, since $\nabla\cdot\mathcal{J}\nabla S = -\nabla\times\nabla S = 0$. Thus, instead of defining

$\vec{r} = \nabla \times \vec{R}$, the incompressible component can be defined as $\vec{r} = \mathcal{J}\nabla R$, where R is a scalar potential. Finally, the vector Laplacian in Equation (2.14) can be simplified as follows:

$$\begin{aligned} \nabla \times \vec{r} &= \nabla \times \vec{V} \\ -\nabla \cdot \mathcal{J}(\vec{r}) &= -\nabla \cdot J\vec{V} \\ -\nabla \cdot \mathcal{J}(\mathcal{J}\nabla R) &= -\nabla \cdot \mathcal{J}\vec{V} \\ \nabla^2 R &= -\nabla \cdot \mathcal{J}\vec{V} \end{aligned}$$

A.1.6 Invalidity of the PN Boundary Conditions

This section provides an analytical example to show that the PN boundary conditions are not applicable to compute the HHD in general, as they may violate the divergence theorem.

Definition A.1 (The HHD with PN boundary conditions)**.** The HHD with PN boundary conditions is a three-component form of the HHD computed using the PN boundary conditions, meaning, its components satisfy the following:

- $\vec{n} \cdot \vec{d} = 0$, that is, the irrotational component is parallel to the boundary.
- $\vec{n} \times \vec{r} = \vec{0}$, that is, the incompressible component is normal to the boundary.

where $\vec{n}$ is the outward normal to the boundary.

To understand the invalidity of the PN boundary conditions, consider a vector field defined by a source at the origin of $\mathbb{R}^2$, that is, $\vec{V}(x, y) = \{x, y\}$. For $\vec{V}$, $\nabla \cdot \vec{V} = 2$, and $\nabla \times \vec{V} = 0$. Since $\vec{V}$ contains no rotation, its irrotational component should be zero, that is, $\vec{r} = 0$. Also, since the field is purely divergent, $\vec{h} = 0$. Thus, computing the HHD for $\vec{V}$ should give $\vec{V} = \vec{d}$.

The divergence of $\vec{d}$ is simply $\nabla \cdot \vec{d} = \nabla \cdot \vec{V} = 2$. Combining the divergence theorem [71] and the PN boundary conditions leads to

$$\int_\Omega \nabla \cdot \vec{d} \, \mathrm{d}A = \int_{\partial\Omega} \vec{n} \cdot \vec{d} \, \mathrm{d}S \tag{A.3}$$

$$\Leftrightarrow \quad \int_\Omega 2 \, \mathrm{d}A = \int_{\partial\Omega} 0 \, \mathrm{d}S \tag{A.4}$$

$$\Leftrightarrow \quad 2\int_\Omega \mathrm{d}A = 0 \tag{A.5}$$

Thus, for this example, the HHD with PN boundary conditions exists only for a domain with zero area. Essentially, by the divergence theorem, any decomposition with PN boundary

conditions has zero global divergence, and, thus, does not apply to most vector fields. More specifically, it can be applied only to the vector fields with

$$\int_{\Omega} \nabla \cdot \vec{V} \ \mathrm{d}A = \int_{\partial\Omega} \vec{n} \cdot \vec{V} = 0.$$

Similarly, it can be shown that for a purely rotational field, $\vec{V}(x, y) = \{-y, x\}$, $\vec{n} \times \vec{r} = \vec{0}$ cannot be maintained. The combined result is that PN boundary conditions are not valid for general vector fields.

A.2 Edge Maps

A.2.1 Derivation of Mapping Error in Edge Maps

The motion of a particle under a linear vector field $\vec{V}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \vec{o}$ is defined as [108]

$$\mathbf{x}(t) = e^{(t-t_0)\mathbf{A}}\mathbf{x}_0 + (e^{(t-t_0)\mathbf{A}} - \mathrm{Id})\mathbf{A}^{-1}\vec{o},$$

where, $\mathbf{x}_0, t_0$ give the particle's initial position and time, and $\mathbf{x}(t)$ gives the position after time t. $\mathbf{A}$ is a 2×2 matrix, $\vec{o}$ is the translation vector, and Id is the identity matrix.

Consider a link, in which origin interval $(\mathbf{a}, \mathbf{b})$ flows to destination interval $(\mathbf{c}, \mathbf{d})$. Point $\mathbf{a}$ flows to point $\mathbf{c}$ in time $t(\mathbf{a})$. Similarly, point $\mathbf{d}$ is calculated as the true destination of point $\mathbf{b}$, reached in time $t(\mathbf{d})$.

$$\mathbf{c}(\mathbf{a}, t(\mathbf{a})) = e^{t(\mathbf{a})\mathbf{A}}\mathbf{a} + (e^{t(\mathbf{a})\mathbf{A}} - \mathrm{Id})\mathbf{A}^{-1}\vec{o}$$
$$\mathbf{d}(\mathbf{b}, t(\mathbf{b})) = e^{t(\mathbf{b})\mathbf{A}}\mathbf{b} + (e^{t(\mathbf{b})\mathbf{A}} - \mathrm{Id})\mathbf{A}^{-1}\vec{o}$$

Now, consider a point $\mathbf{x}$ on the source interval, whose true destination is given by $\mathbf{y}$.

$$\mathbf{y}(\mathbf{x}, t(\mathbf{x})) = e^{t(\mathbf{x})\mathbf{A}}\mathbf{x} + (e^{t(\mathbf{x})\mathbf{A}} - \mathrm{Id})\mathbf{A}^{-1}\vec{o}$$

Suppose, $\mathbf{x} = \lambda\mathbf{a} + (1 - \lambda)\mathbf{b}$. We can interchangebly use $\mathbf{y}(\mathbf{x})$ and $\mathbf{y}(\lambda)$, and $t(\mathbf{x})$ and $t(\lambda)$.

The map gives $\mathbf{x}'$s destination as $\mathbf{y}'$, and the mapping error is calculated as $e(\lambda) = \mathbf{y}'(\lambda) - \mathbf{y}(\lambda)$. The map interpolates the destination interval to approximate the destination of $\mathbf{x}$.

$$\begin{aligned}\mathbf{y}'(\lambda) =& \lambda\mathbf{c} + (1 - \lambda)\mathbf{d} \\ =& \lambda e^{t(\mathbf{a})\mathbf{A}}(\mathbf{a} + \mathbf{A}^{-1}\vec{o}) + (1 - \lambda)e^{t(\mathbf{b})\mathbf{A}}(\mathbf{b} + \mathbf{A}^{-1}\vec{o}) - \mathrm{Id}\ \ \mathbf{A}^{-1}\vec{o}\end{aligned}$$

Calculating the deviation between the two,

$$\begin{aligned} e(\lambda) &= \mathbf{y}'(\lambda) - \mathbf{y}(\lambda) \\ &= \lambda e^{t(\mathbf{a})\mathbf{A}}(\mathbf{a} + \mathbf{A}^{-1}\vec{o}) + (1-\lambda)e^{t(\mathbf{b})\mathbf{A}}(\mathbf{b} + \mathbf{A}^{-1}\vec{o}) - \mathrm{Id}\ \ \mathbf{A}^{-1}\vec{o} \\ &\qquad\qquad - \ \{e^{t(\mathbf{x})\mathbf{A}}\mathbf{x} + (e^{t(\mathbf{x})\mathbf{A}} - \mathrm{Id})\mathbf{A}^{-1}\vec{o}\} \\ &= \lambda e^{t(\mathbf{a})\mathbf{A}}(\mathbf{a} + \mathbf{A}^{-1}\vec{o}) + (1-\lambda)e^{t(\mathbf{b})\mathbf{A}}(\mathbf{b} + \mathbf{A}^{-1}\vec{o}) - e^{t(\mathbf{x})\mathbf{A}}(\mathbf{x} + \mathbf{A}^{-1}\vec{o}) \\ &= \lambda e^{t(\mathbf{a})\mathbf{A}}(\mathbf{a} + \mathbf{A}^{-1}\vec{o}) + (1-\lambda)e^{t(\mathbf{b})\mathbf{A}}(\mathbf{b} + \mathbf{A}^{-1}\vec{o}) - \ e^{t(\lambda)\mathbf{A}}(\lambda\mathbf{a} + (1-\lambda)\mathbf{b} + \mathbf{A}^{-1}\vec{o}) \end{aligned} \tag{A.6}$$

The maximum length of this deviation of $e(\lambda)$ over every link is assigned as the mapping error of the link

$$\varepsilon = \max_{0 \le \lambda \le 1} \ ||e(\lambda)||_2$$

APPENDIX B

DATASETS

B.1 2D Flows

B.1.1 Flow Inside the Homogeneous Charge Compression Ignition Engine

The flow visualized in Figure B.1 is a 2D flow resulting from the simulation of a homogeneous charge compression ignition (HCCI) [88]. The HCCI engines have higher compression ratio and no throttling losses, and, therefore, they may be able to provide efficiency gains over conventional spark-ignition engines. The flow was generated by the direct numerical simulation (DNS) of autoignition performed by Dr. Jacqueline Chen at the Sandia National Laboratories. This 2D flow is incompressible, and exhibits turbulent vortical structures of clockwise and counterclockwise rotations. It is defined on a $[640 \times 640]$ domain that is periodic in both horizontal and vertical dimensions (diffeomorphic to a torus). The flow is unsteady, and is available for 299 time-steps.

B.1.2 Ocean Winds Flow

The flow illustrated in Figure B.2 represents a simulation of the wind flow on the ocean's surface. This 2D incompressible flow is defined on a $[573 \times 288]$ regular grid, and is due to Zhanping Liu at the Mississippi State University. We thank Han-Wei Shen of the Ohio State University for providing us access to this data.

B.1.3 Flow Behind a Cylinder

Figure B.3 illustrates a well-studied flow phenomenon representing the flow moving past a rigid obstacle, a cylinder in this case. The characteristic feature of this phenomenon is the shedding of vortices in the wake of the cylinder, called the von Kármán vortex street. The von Kármán vortex street is a sequence of counter-rotating vortices shed alternately by the top and the bottom of the cylinder. The shedding phenomenon is found in nature, for example, winds blowing past mountains or isolated islands, in engineering problems, such as

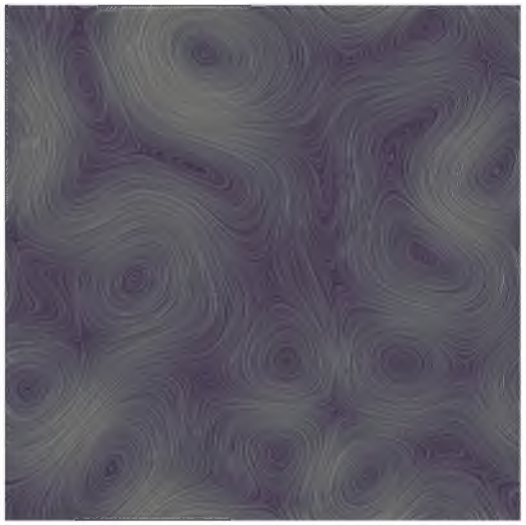


Figure B.1. Visualization of the flow inside a homogeneous charge compression ignition engine (HCCI) engine. Lighter color represents faster velocity.

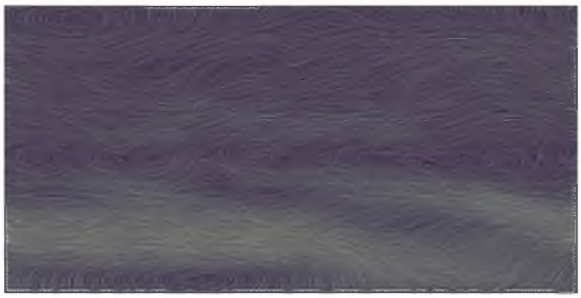


Figure B.2. Visualization of the ocean winds flow. Lighter color represents faster velocity.

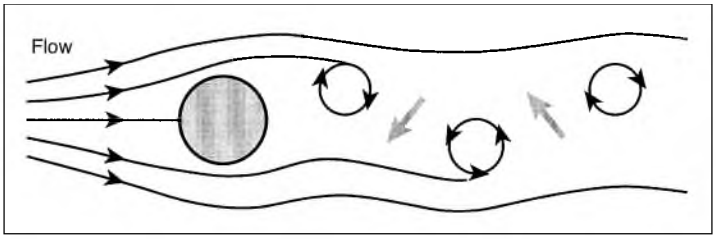


Figure B.3. Illustration of the von Kármán vortex street in the flow behind a cylinder [45]. Notice the backward flow between counter-rotating vortices that induces drag on the cylinder.

design of skyscrapers, etc. In particular, the collapse of the three towers of the Ferrybridge Power Station C in 1965 was caused by the vortex shedding created during high winds.

The properties of the shedding, such as its speed and frequency, depend upon the velocity and the viscosity of the flow, and the size of the cylinder – all of which are characterized by the Reynolds number (Re). This flow is common in the fluids and analysis community, and is often used to illustrate vortex detection techniques. The 2D flow behind a cylinder used in this dissertation was simulated by Dr. Mahsa Mirzargar at the SCI Institute using Nektar++ [23]. The flow is sampled on a $[1450 \times 800]$ domain for various Reynolds number and time ranges.

B.2 Surface Flows

B.2.1 Interface Surface in Rayleigh-Taylor Instability

Laney et al. [115] performed a topological analysis on the interface surface of a heavy fluid placed above a light fluid to understand the Rayleigh-Taylor instability. The interface surface shows a characteristic structure of rising bubbles and falling spikes, as visualized in Figure B.4. The data was generated by William Cabot and Andrew Cook at the Lawrence Livermore National Laboratory. The results presented in this dissertation use the gradient field of the height function defined on the interface surface, and focus on an interesting subregion containing $27,737$ vertices and $54,737$ triangles.

Figure B.4. A topological segmentation of the height function of the bubbles surface representing the Rayleigh-Taylor instability.

B.2.2 Surface Flows of Combustion Chamber and Diesel Engine

Figure B.5 shows visualizations of simulated flows on the surface of a combustion chamber and a diesel engine. The goal of studying these flows is to achieve ideal flow patterns inside the engines to optimize the mixture of oxygen and fuel during the ignition phase of the valve cycle, ultimately leading to more efficient combustion. The surface of the combustion chamber consists of $13,151$ vertices and $26,298$ triangles, whereas that of the diesel engine consists of $110,789$ vertices and $221,574$ triangles. We thank Robert Laramee for giving us access to these datasets.

B.3 3D Flows

B.3.1 Lifted Ethylene Jet Flame

Figure B.6 shows a 2D slice through the center of the flow generated by a large-scale combustion simulation of a lifted ethylene jet flame [223, 222] performed using S3D [29]. Developed at the Sandia National Laboratories, S3D is designed to perform massively parallel direct numerical simulation for turbulent combustion, which uses a Runge-Kutta integrator to solve the full compressible Navier-Stokes equation, and balance equations for conserving mass, momentum, energy, and chemically reactive species. The resulting flow is used to

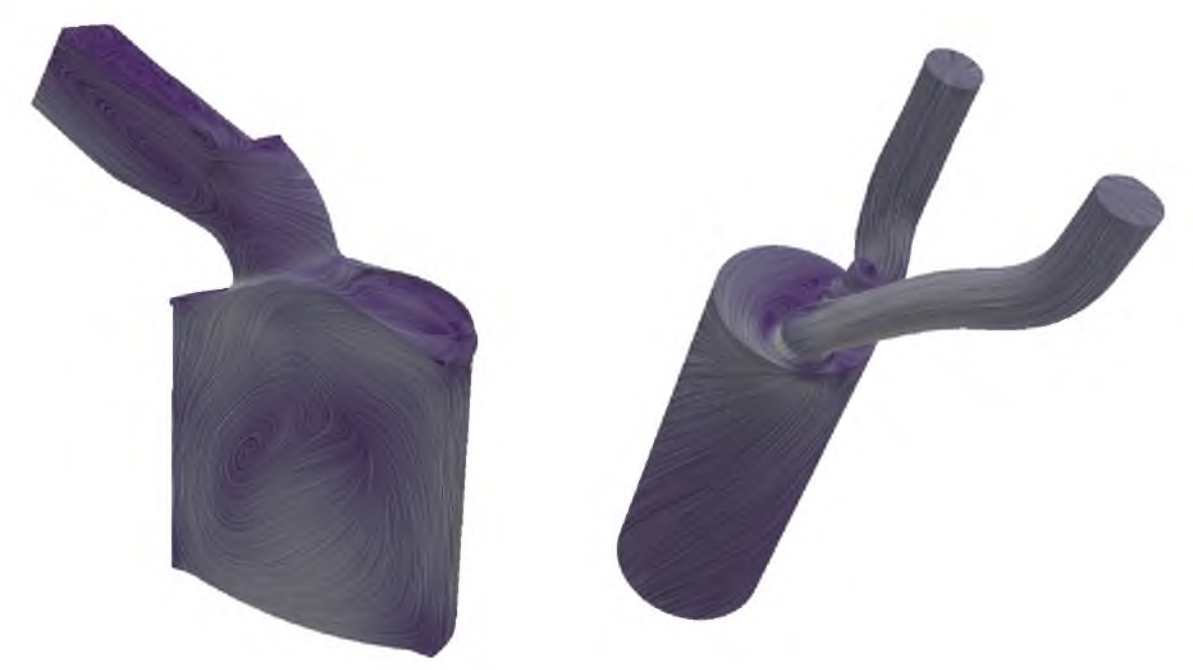


Figure B.5. Visualization of the surface flows on (a) a combustion chamber and (b) a diesel engine. Lighter color represents faster velocity.

Figure B.6. Visualization of a 2D slices through the center of the 3D lifted ethylene jet flame. Lighter color represents faster velocity.

investigate turbulent lifted jet flames with the goal of improving the understanding of direct injection stratified spark ignition engines for commercial boilers, as well as fundamental combustion phenomena.

The flow in consideration is compressible, and is defined on a $[2025 \times 1600 \times 400]$ rectilinear grid, with a variable spacing in the horizontal dimension, and is available for a total of 299 time-steps. The data was generated on Jaguar (now Titan) at the Oak Ridge leadership computing facility, using $30,000$ cores with $[27 \times 40 \times 40]$ grid points per processor. The simulation captures a large number of variables representing the flow velocity, temperature, pressure, species, etc., and the total file-size for storing these variables, just for a single time-step, is about 280 GB.

B.3.2 Jet in Cross-Flow

The flow illustrated in Figure B.7 represents a jet in cross-flow (JICF) [73], which is a fundamental flow phenomenon relevant to many engineering applications, e.g., film cooling of turbines, fuel injections, and dilution jets in gas-turbine combustors. The experimental set-up contains injection of flow through a jet at the bottom in the presence of a strong background flow in transverse direction, the cross-flow. The goal of the experiment is to study different types of vortical structures created by the interaction of the burning jet with the cross-flow. The most prominent structures in this flow are a pair of counter-rotating

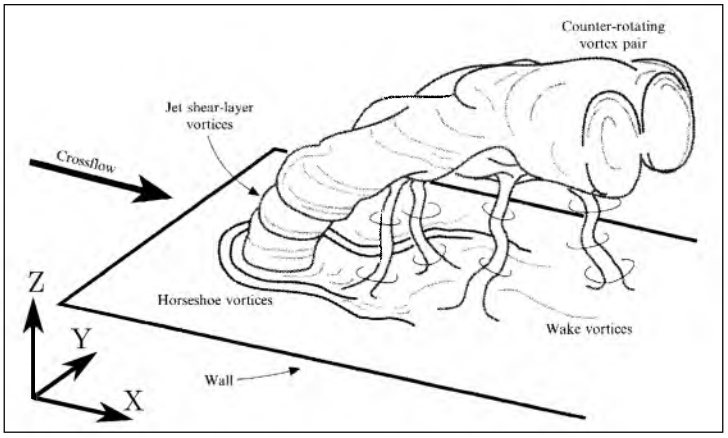


Figure B.7. Illustration [60] of the four types of vortical structures associated with this flow: jet shear-layer vortices at the perimeter of the bending jet, the developing counter-rotating vortex pair, horseshoe vortices on the wall, and wake vortices extending from the wall to the jet. Reproduced with permission from Cambridge University Press.

vortices, which occur as a result of the impulse of the jet on the cross-flow, and become dominant in the far field. At the periphery of the jets inflow, jet shear-layer vortices are created due to the annular shear layer that separates from the edge of the jet orifice. These vortices dominate the initial portion of the jet. The horseshoe vortices wrap around the base of the jet issuing from a wall into a cross-flow, and the wake vortices are the structures existing between the bottom wall and the jet itself. The JICF data used in this work was simulated by Dr. Jacqueline Chen at the Sandia National Laboratories. This 3D flow is incompressible, and is defined on a $[2025 \times 1600 \times 400]$ regular grid.

B.3.3 Global Oceanic Currents

Figure B.8 shows a visualization of the top slice of a 3D simulation of global oceanic currents. These simulations were produced using a technique based on boundary impulse response functions [133] with the aim of studying the global eddies patterns in oceanic currents. The data is available on a $[3600 \times 2400 \times 300]$ regular grid, and 365 time-steps representing each day of an year. Although the flow in the original 3D simulation is incompressible, a 2D depth slice from this 3D flown does not satisfy incompressibility anymore, since slicing creates sinks and sources due to the flow across depths. We thank Mathew Maltude from the Climate, Ocean and Sea Ice Modelling program at Los Alamos National Laboratory and the Biological and Environment (BER) Office of Science and the Ultrascale Visualization - Climate Data Analysis Tool (UV-CDAT) team for providing us access to this data.

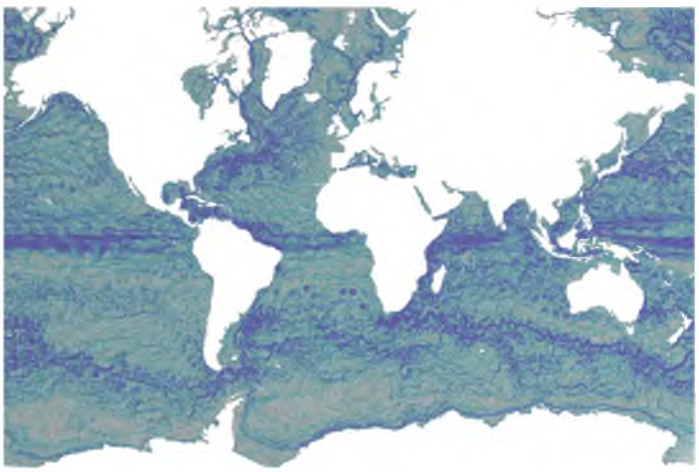


Figure B.8. Visualization of the top 2D slice of 3D global oceanic currents. Darker color represents faster flow.

REFERENCES

[1] Abraham, R., Marsden, J. E., and Ratiu, T. *Manifolds, Tensor Analysis and Applications*, 2nd ed., vol. 75 of *Applied Mathematical Science*. Springer, 1988.

[2] Ahusborde, E., Azaiez, M., and Caltagirone, J.-P. A primal formulation for the Helmholtz decomposition. *Journal of Computational Physics 225*, 1 (2007), 13–19.

[3] Aki, K., and Richards, P. G. *Quantitative Seismology*, 2nd ed. University Science Books, 2002.

[4] Akram, M., and Michel, V. Regularisation of the Helmholtz decomposition and its application to geomagnetic field modelling. *GEM – International Journal on Geomathematics 1*, 1 (2010), 101–120.

[5] Amrouche, C., Bernardi, C., Dauge, M., and Girault, V. Vector potentials in three-dimensional nonsmooth domains. *Mathematical Methods in the Applied Sciences 21*, 9 (1998), 823–864.

[6] Aref, H., Rott, N., and Thomann, H. Gröbli's solution of the three-vortex problem. *Annual Review of Fluid Mechanics 24* (1992), 1–21.

[7] Arfken, G. *Mathematical Methods for Physicists*, 2nd ed. Academic, New York, 1970.

[8] Auchmuty, G. Potential representation of incompressible vector fields. In *Nonlinear Problems in Applied Mathematics*, T. S. Angell, L. P. Cook, R. E. Kleinman, and W. E. Olmstead, Eds. SIAM, 1995, pp. 43–49.

[9] Barnat, J., and Moravec, P. Parallel algorithms for finding SCCs in implicitly given graphs. In *Formal Methods: Applications and Technology*, L. Brim, B. Haverkort, M. Leucker, and J. van de Pol, Eds., vol. 4346 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 316–330.

[10] Batra, R., and Hesselink, L. Feature comparisons of 3-D vector fields using earth mover's distance. In *Proceedings of IEEE Visualization* (1999), pp. 105–114.

[11] Bell, J. B., Colella, P., and Glaz, H. M. A second order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics 85*, 2 (1989), 257–283.

[12] Bell, J. B., and Marcus, D. L. A second-order projection method for variable-density flows. *Journal of Computational Physics 101*, 2 (1992), 334–348.

[13] Bonneau, G.-P., Hege, H.-C., Johnson, C. R., Oliveira, M. M., Potter, K., Rheingans, P., and Schultz, T. Overview and state-of-the-art of uncertainty visualization. In *Scientific Visualization – Uncertainty, Multifield, Biomedical, and Scalable Visualization*, C. D. Hansen, M. Chen, C. R. Johnson, A. E. Kaufman, and H. Hagen, Eds., Mathematics and Visualization. Springer London, 2014, pp. 3–27.

[14] Braun, H., and Hauck, A. Tomographic reconstruction of vector fields. *IEEE Transactions on Signal Processing 39*, 2 (1991), 464–471.

[15] Braun, S. A., Montgomery, M. T., and Pu, Z. High-resolution simulation of hurricane bonnie (1998). part I: The organization of eyewall vertical motion. *Journal of the Atmospheric Sciences 63*, 1 (2006), 19–42.

[16] Bresenham, J. E. Algorithm for computer control of a digital plotter. *IBM Systems Journal 4*, 1 (1965), 25–30.

[17] Brown, D. L., Cortez, R., and Minion, M. L. Accurate projection methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics 168*, 2 (2001), 464–499.

[18] Brown, G. L., and Roshko, A. On density effects and large structures in turbulent mixing layers. *Journal of Fluid Mechanics 64* (1974), 775–816.

[19] Bykhovskiy, E. B., and Smirnov, N. V. *On Orthogonal Expansions of the Space of Vector Functions Which are Square-Summable over a Given Domain and the Vector Analysis Operators*. Academy of Sciences USSR Press, 1960.

[20] Cabral, B., and Leedom, L. C. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), SIGGRAPH '93, pp. 263–270.

[21] Cacuci, D. G. *Sensitivity and Uncertainty Analysis: Theory*, vol. I. Chapman & Hall/CRC, Boca Raton, 2003.

[22] Caltagirone, J.-P., and Breil, J. A vector projection method for solving the Navier-Stokes equations. *Comptes Rendus de l'Académie des Sciences - Series IIB – Mechanics-Physics-Astronomy 327*, 11 (1999), 1179–1184.

[23] Cantwell, C. D., Moxey, D., Comerford, A., Bolis, A., Rocco, G., Mengaldo, G., Grazia, D. d., Yakovlev, S., Lombard, J. E., Ekelschot, D., Jordi, B., Xu, H., Mohamied, Y., Eskilsson, C., Nelson, B., Vos, P., Biotto, C., Kirby, R. M., and Sherwin, S. J. Nektar++: An open-source spectral/hp element framework. *Computer Physics Communications* (Feb. 2015).

[24] Chakraborty, P., Balachandar, S., and Adrian, R. J. On the relationships between local vortex identification schemes. *Journal of Fluid Mechanics 535* (2005), 180–214.

[25] Chen, G., Deng, Q., Szymczak, A., Laramee, R. S., and Zhang, E. Morse set classification and hierarchical refinement using Conley index. *IEEE Transactions on Visualization and Computer Graphics 18*, 5 (2012), 767–782.

[26] Chen, G., Mischaikow, K., Laramee, R. S., Pilarczyk, P., and Zhang, E. Vector field editing and periodic orbit extraction using Morse decomposition. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 769–785.

[27] Chen, G., Mischaikow, K., Laramee, R. S., and Zhang, E. Efficient Morse decompositions of vector fields. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 848–862.

[28] Chen, G., Palke, D., Lin, Z., Yeh, H., Vincent, P., Laramee, R. S., and Zhang, E. Asymmetric tensor field visualization for surfaces. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1979–1988.

[29] Chen, J. H., Choudhary, A., de Supinski, B., DeVries, M., Hawkes, E. R., Klasky, S., Liao, W. K., Ma, K.-L., Mellor-Crummey, J., Podhorszki, N., Sankaran, R., Shende, S., and Yoo, C. S. Terascale direct numerical simulations of turbulent combustion using S3D. *Computational Science & Discovery 2*, 1 (2009).

[30] Chong, M. S., Perry, A. E., and Cantwell, B. J. A general classification of three-dimensional flow fields. *Physics of Fluids A 2*, 5 (1990), 408–420.

[31] Chorin, A. J. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics 2*, 1 (1967), 12–26.

[32] Chorin, A. J. Numerical solution of the Navier-Stokes equations. *Mathematics of Computations 22* (1968), 745–762.

[33] Chorin, A. J. On the convergence of discrete approximations to the Navier-Stokes equations. *Mathematics of Computations 23* (1969), 341–353.

[34] Chorin, A. J., and Marsden, J. E. *A Mathematical Introduction to Fluid Mechanics*. Springer, 1993.

[35] Colin, F., Egli, R., and Lin, F. Y. Computing a null divergence velocity field using smoothed particle hydrodynamics. *Journal of Computational Physics 217*, 2 (2006), 680–692.

[36] Courant, R., and Hilbert, D. *Methods of Mathematical Physics*. Publisher Inc.: New York, 1953.

[37] Courant, R., Isaacson, E., and Rees, M. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communication on Pure and Applied Mathematics 5*, 3 (1952), 243–255.

[38] Cummins, S. J., and Rudman, M. An SPH projection method. *Journal of Computational Physics 152*, 2 (1999), 584–607.

[39] de Leeuw, W., and van Liere, R. Collapsing flow topology using area metrics. In *Proceedings of IEEE Visualization* (1999), pp. 349–354.

[40] Denaro, F. M. On the application of the Helmholtz-Hodge decomposition in projection methods for incompressible flows with general boundary conditions. *International Journal for Numerical Methods in Fluids 43*, 1 (2003), 43–69.

[41] Deriaz, E., and Perrier, V. Divergence-free and curl-free wavelets in 2D and 3D, application to turbulent flows. *Journal of Turbulence 7*, 3 (2006), 1–37.

[42] Deriaz, E., and Perrier, V. Direct numerical simulation of turbulence using divergence-free wavelets. *Multiscale Modeling and Simulation 7*, 3 (2008), 1101–1129.

[43] Deriaz, E., and Perrier, V. Orthogonal Helmholtz decomposition in arbitrary dimension using divergence-free and curl-free wavelets. *Applied and Computational Harmonic Analysis 26*, 2 (2009), 249–269.

[44] Dinh, H. Q., and Xu, L. Measuring the similarity of vector fields using global distributions. In *Structural, Syntactic, and Statistical Pattern Recognition*, N. da Vitoria Lobo, T. Kasparis, F. Roli, J. T. Kwok, M. Georgiopoulos, G. C. Anagnostopoulos, and M. Loog, Eds., vol. 5342 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 187–196.

[45] Drucker, E. G., and Lauder, G. V. Experimental hydrodynamics of fish locomotion: Functional insights from wake visualization. *Integrative and Comparative Biololgy 42*, 2 (2002), 243–257.

[46] E, W., and Liu, J.-G. Projection method I: Convergence and numerical boundary layers. *SIAM Journal on Numerical Analysis 32*, 4 (1995), 1017–1057.

[47] Edelsbrunner, H., and Harer, J. *Computational Topology: An Introduction*. American Mathematical Society, 2009.

[48] Edelsbrunner, H., Harer, J., and Zomorodian, A. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry 30*, 1 (2003), 87–107.

[49] Edelsbrunner, H., and Mücke, E. P. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics 9*, 1 (1990), 66–104.

[50] Elber, G., and Kim, M.-S. Geometric constraint solver using multivariate rational spline functions. In *Proceedings of the 6th ACM Symposium on Solid Modeling and Applications* (2001), pp. 1–10.

[51] Fedkiw, R., Stam, J., and Jensen, H. W. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), SIGGRAPH '01, pp. 15–22.

[52] Fisher, M., Schröder, P., Desbrun, M., and Hoppe, H. Design of tangent vector fields. *ACM Transactions on Graphics 26*, 3 (2007).

[53] Floater, M. S., Kós, G., and Reimers, M. Mean value coordinates in 3D. *Computer Aided Geometric Design 22*, 7 (2005), 623–631.

[54] Forman, R. A discrete Morse theory for cell complexes. In *Geometry, Topology and Physics for Raoul Bott*, S.-T. Yau, Ed. International Press of Boston, Inc., 1995.

[55] Forman, R. Combinatorial vector fields and dynamical systems. *Mathematische Zeitschrift 228*, 4 (1998), 629–681.

[56] Forman, R. Morse theory for cell complexes. *Advances in Mathematics 134*, 1 (1998), 90–145.

[57] Forman, R. A user's guide to discrete Morse theory. In *Proceedings of the 2001 International Conference on Formal Power Series and Algebraic Combinatorics* (2001).

[58] Foster, N., and Metaxas, D. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483.

[59] Foster, N., and Metaxas, D. Modeling the motion of a hot, turbulent gas. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), SIGGRAPH '97, pp. 181–188.

[60] Frik, T. F., and Roshko, A. Vortical structure in the wake of a transverse jet. *Journal of Fluid Mechanics 279* (1994), 1–47.

[61] Fuchs, R., Kemmler, J., Schindler, B., Waser, J., Sadlo, F., Hauser, H., and Peikert, R. Toward a Lagrangian vector field topology. *Computer Graphics Forum 29*, 3 (2010), 1163–1172.

[62] Gao, H., Mandal, M. K., Guo, G., and Wan, J. Singular point detection using discrete Hodge Helmholtz decomposition in fingerprint images. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)* (2010), pp. 1094 –1097.

[63] Garth, C., F, G., Tricoche, X., and Hagen, H. Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 1464–1471.

[64] Garth, C., Li, G.-S., Tricoche, X., Hansen, C. D., and Hagen, H. Visualizing Lagrangian coherent structures and comparison to vector field topology. In *Topology-Based Methods in Visualization II*, H.-C. Hege, K. Polthier, and G. Scheuermann, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 1–13.

[65] Garth, C., and Tricoche, X. Topology- and feature-based flow visualization: Methods and applications. In *SIAM Conference on Geometric Design and Computing* (2005).

[66] Garth, C., Tricoche, X., Salzbrunn, T., and Scheuermann, G. Surface techniques for vortex visualization. In *Proceedings of Eurographics / IEEE TCVG Symposium on Visualization (VisSym)* (2004), pp. 155–164.

[67] Georgobiani, D., Mansour, N. N., Kosovichev, A. G., Stein, R. F., and Å Nordlund. Velocity field decomposition in 3D numerical simulations of solar turbulent convection. *NASA Center for Turbulence Research - Annual Research Briefs* (2004), 355–340.

[68] Girault, V. Incompressible finite element methods for Navier-Stokes equations with nonstandard boundary conditions in R^3. *Mathematics of Computations 51*, 183 (1988), 55–74.

[69] Globus, A., Levit, C., and Lasinski, T. A tool for visualizing the topology of three-dimensional vector fields. In *Proceedings of IEEE Conference on Visualization* (1991), pp. 33–41.

[70] Goldberg, D. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys 23*, 1 (1991), 5–48.

[71] Griffiths, D. J. *Introduction to Electrodynamics*, 2nd ed. Addison Wesley, 1999.

[72] Grinspun, E., and Secord, A. Introduction to discrete differential geometry: the geometry of plane curves. In *ACM SIGGRAPH ASIA 2008 courses* (2008), ACM, pp. 9:1–9:4.

[73] Grout, R. W., Gruber, A., Kolla, H., Bremer, P.-T., Bennett, J. C., Gyulassy, A. G., and Chen, J. H. A direct numerical simulation study of turbulence and flame structure in transverse jets analysed in jet-trajectory based coordinates. *Journal of Fluid Mechanics 706* (2012), 351–383.

[74] Guo, Q. Cardiac video analysis using the Hodge Helmholtz field decomposition. Master's thesis, Department of Electrical and Computer Engineering, University of Alberta, 2004.

[75] Guo, Q., Mandal, M. K., and Li, M. Y. Efficient Hodge-Helmholtz decomposition of motion fields. *Pattern Recognition Letters 26*, 4 (2005), 493–501.

[76] Guo, Q., Mandal, M. K., Liu, G., and Kavanagh, K. M. Cardiac video analysis using Hodge-Helmholtz field decomposition. *Computers in Biology and Medicine 36*, 1 (2006), 1–20.

[77] Gyulassy, A. G. *Combinatorial Construction of Morse-Smale Complexes for Data Analysis and VIsualization.* PhD thesis, University of California, Davis, 2008.

[78] Gyulassy, A. G., Natarajan, V., Pascucci, V., and Hamann, B. Efficient computation of Morse-Smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1440–1447.

[79] Hairer, E., Norsett, S. P., and Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer Series in Computational Mathematics, 1993.

[80] Haller, G. Finding finite-time invariant manifolds in two-dimensional velocity fields. *Chaos 10*, 1 (2000), 99–108.

[81] Haller, G. Lagrangian coherent structures and the rate of strain in two-dimensional turbulence. *Physics of Fluids A 13* (2001), 3365–3385.

[82] Haller, G. An objective definition of a vortex. *Journal of Fluid Mechanics 525* (2005), 1–26.

[83] Haller, G. A variational theory of hyperbolic Lagrangian coherent structures. *Physica D 240* (2011), 574–598.

[84] Harlow, F. H., and Welch, J. E. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids 8*, 12 (1965), 2182–2189.

[85] Hatton, R. L., and Choset, H. Optimizing coordinate choice for locomoting systems. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (2010), pp. 4493–4498.

[86] Hauser, W. On the fundamental equations of electromagnetism. *American Journal of Physics 38*, 1 (1970), 80–85.

[87] Hauser, W. *Introduction to the Principles of Electromagnetism.* Addison-Wesley Educational Publishers Inc, 1971.

[88] Hawkes, E. R., Sankaran, R., Pébay, P. P., and Chen, J. H. Direct numerical simulation of ignition front propagation in a constant volume with temperature inhomogeneities: II. Parametric study. *Combustion and Flame 145*, 1–2 (2006), 145–159.

[89] Helman, J. L., and Hesselink, L. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer 22*, 8 (1989), 27–36.

[90] Helman, J. L., and Hesselink, L. Surface representations of two- and three-dimensional fluid flow topology. In *Proceedings of IEEE Conference on Visualization* (1990), pp. 6–13.

[91] Helman, J. L., and Hesselink, L. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications 11*, 3 (1991), 34–46.

[92] Henle, M. *A Combinatorial Introduction to Topology.* Dover Publications, Inc., New York, NY, USA, 1994.

[93] Hinkle, J., Fletcher, P. T., Wang, B., Salter, B., and Joshi, S. 4D MAP image reconstruction incorporating organ motion. In *Information Processing in Medical Imaging*, J. L. Prince, D. L. Pham, and K. J. Myers, Eds., Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 676–687.

[94] Hlawatsch, M., Leube, P., Wolfgang, N., and Weiskopf, D. Flow radar glyphs–static visualization of unsteady flow with uncertainty. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1949–1958.

[95] Hodge, W. V. D. *The Theory and Applications of Harmonic Integrals.* Cambridge University Press: Cambridge, 1952.

[96] Hultquist, J. P. M. Constructing stream surfaces in steady 3D vector fields. In *Proceedings of IEEE Conference on Visualization* (1992), pp. 171–178.

[97] Hunt, J. C. R. Vorticity and vortex dynamics in complex turbulent flows. *Transactions of Canadian Society for Mechanical Engineering 11*, 1 (1987), 21–35.

[98] Hussain, A. K. M. F. Coherent structures — reality and myth. *Physics of Fluids 26* (1983).

[99] Ingber, M. S., and Kempka, S. N. A Galerkin implementation of the generalized Helmholtz decomposition for vorticity formulations. *Journal of Computational Physics 169*, 1 (2001), 215–237.

[100] Jeong, J., and Hussain, F. On the identification of a vortex. *Journal of Fluid Mechanics 285* (1995), 69–94.

[101] Johnson, C. R. Top scientific visualization research problems. *IEEE Computer Graphics and Applications 24*, 4 (2004), 13–17.

[102] Johnson, C. R., and Sanderson, A. R. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications 23*, 5 (2003), 6–10.

[103] Johnson, S. A., Greenleaf, J. F., Tanaka, M., and Flandro, G. *Acoustical Holography*, vol. 7. Plenum, New York, 1977.

[104] Joseph, D. D. Helmholtz decomposition coupling rotational to irrotational flow of a viscous fluid. *PNAS 103*, 39 (2006), 14272–14277.

[105] Kasten, J., Reininghaus, J., Hotz, I., and Hege, H.-C. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2080–2087.

[106] Kenwright, D. N., Henze, C., and Levit, C. Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 135–144.

[107] Kim, J., and Moin, P. Application of a fractional-step method to incompressible Navier-Stokes equations. *Journal of Computational Physics 59*, 2 (1985), 308–323.

[108] Kipfer, P., Reck, F., and Greiner, G. Local exact particle tracing on unstructured grids. *Computer Graphics Forum 22*, 2 (2003), 133–142.

[109] Kobe, D. H. Helmholtz theorem for antisymmetric second-rank tensor fields and electromagnetism with magnetic monopoles. *American Journal of Physics 52*, 4 (1984), 354–358.

[110] Kobe, D. H. Helmholtz's theorem revisited. *American Journal of Physics 54*, 6 (1986), 552–554.

[111] Kodaira, K. Harmonic fields in Riemannian manifolds. *Annals of Mathamatics 50*, 3 (1949), 587–665.

[112] Kumar, A., and Barve, S. *How and Why In Basic Mechanics.* Universities Press, 2003.

[113] Ladyzhenskaja, O. A. *The Mathematical Theory of Viscous Incompressible Flow.* Gordon and Breach: New York, 1963.

[114] Lamb, H. *Hydrodynamics*, 6th ed. Cambridge University Press, 1932.

[115] Laney, D. E., Bremer, P.-T., Mascarenhas, A., Miller, P., and Pascucci, V. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1053–1060.

[116] Laramee, R. S., Hauser, H., Zhao, L., and Post, F. H. Topology-based flow visualization: The state of the art. In *Topology-based Methods in Visualization*, H. Hauser, H. Hagen, and H. Theisel, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2007, pp. 1–19.

[117] Laramee, R. S., Weiskopf, D., Schneider, J., and Hauser, H. Investigating swirl and tumble flow with a comparison of visualization techniques. In *Proceedings of IEEE Visualization* (2004), pp. 51–58.

[118] Lavin, Y., Batra, R., and Hesselink, L. Feature comparisons of vector fields using earth mover's distance. In *Proceedings of IEEE Visualization* (1998), pp. 103–109.

[119] Lekien, F., Coulliette, C., Bank, R., and Marsden, J. E. Open-boundary modal analysis: Interpolation, extrapolation, and filtering. *Journal of Geophysical Research 109*, C12004 (2004).

[120] Lewiner, T., Lopes, H., and Tavares, G. Visualizing forman s discrete vector field. In *Visualization and Mathematics III (VisMath 2002)*, H.-C. Hege and K. Polthier, Eds. Springer Heidelberg, 2002, pp. 95–112.

[121] Lewiner, T., Lopes, H., and Tavares, G. Applications of Forman s discrete Morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics 10*, 5 (2004), 499–508.

[122] Li, W.-C., Vallet, B., Ray, N., and Lévy, B. Representing higher-order singularities in vector fields on piecewise linear surfaces. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1315–1322.

[123] Lloyd, N. G. *Degree theory.* Cambridge University Press, 1978.

[124] Lodha, S. K., Pang, A. T., Sheehan, R. E., and Wittenbrink, C. M. UFLOW: Visualizing uncertainty in fluid flow. In *Proceedings of IEEE Visualization* (1996), pp. 249–254.

[125] Lodha, S. K., Renteria, J. C., and Roskin, K. M. Topology preserving compression of 2D vector fields. In *Proceedings of the IEEE Conference on Visualization* (2000), pp. 343–350.

[126] Lorenzi, M., Ayache, N., and Pennec, X. Regional flux analysis of longitudinal atrophy in Alzheimer's disease. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012*, N. Ayache, H. Delingette, P. Golland, and K. Mori, Eds., vol. 7510 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 739–746.

[127] Losasso, F., Gibou, F., and Fedkiw, R. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 23*, 3 (2004), 457–462.

[128] Lucy, L. B. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal 82* (1977), 1013–1024.

[129] Lugt, H. J. The dilemma of defining a vortex. In *Recent Developments in Theoretical and Experimental Fluids Mechanics*, U. Müller, K. G. Roesner, and B. Schmidt, Eds. Springer Berlin Heidelberg, 1979, pp. 309–321.

[130] Luo, C., Safa, I., and Wang, Y. Feature-aware streamline generation of planar vector fields via topological methods. *Computers & Graphics 36*, 6 (2012), 754–766.

[131] Macêdo, I., and Castro, R. Learning divergence-free and curl-free vector fields with matrix-valued kernels. Tech. rep., Instituto de Matemática Pura e Aplicada, Rio de Janeiro, Brazil, 2010.

[132] Mahrous, K., Bennett, J. C., Scheuermann, G., Hamann, B., and Joy, K. I. Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004), 198–205.

[133] Maltrud, M., Bryan, F., and Peacock, S. Boundary impulse response functions in a century-long eddying global ocean simulation. *Environmental Fluid Mechanics 10*, 1–2 (2010), 275–295.

[134] Mansour, N. N., Kosovichev, A. G., Georgobiani, D., Wray, A., and Miesch, M. Turbulence convection and oscillations in the Sun. In *Proceedings of the SOHO14/GONG 2004 Workshop, "Helio- and Astero-seismology: Towards a Golden Future", ESA SP-559.* (2004).

[135] Margenau, H., and Murphy, G. M. *The Mathematics of Physics and Chemisty.* D Van Nostrand Company, Inc, 1943.

[136] Mascarenhas, A., Grout, R. W., Bremer, P.-T., Hawkes, E. R., Pascucci, V., and Chen, J. H. Topological feature extraction for comparison of terascale combustion simulation data. In *Topological Methods in Data Analysis and Visualization – Theory, Algorithms, and Applications*, V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2011, pp. 229–240.

[137] Matsumoto, Y. *An Introduction to Morse Theory.* American Mathematical Society, 2002. Translated from Japanese by Kiki Hudson and Masahico Saito.

[138] McDonough, J. M. Lectures in elementary fluid dynamics. University of Kentucky, Lexington, 1987.

[139] Miller, B. P. Interpretations from Helmholtz theorem in classical electromagnetism. *American Journal of Physics 52* (1984), 948.

[140] Milnor, J. *Morse Theory.* Princeton University Press, 1963.

[141] Milnor, J. *Lectures on the h-cobordism theorem, Princeton Mathematical Notes.* Princeton University Press, 1965.

[142] Min, C., and Gibou, F. A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids. *Journal of Computational Physics 219*, 2 (2006), 912–929.

[143] Min, C., Gibou, F., and Ceniceros, H. D. A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids. *Journal of Computational Physics 218*, 1 (2006).

[144] Mochizuk, Y., and Imiya, A. Spatial reasoning for robot navigation using the Helmholtz-Hodge decomposition of omnidirectional optical flow. In *Proceedings of 24th International Conference on Image and Vision Computing (IVCNZ)* (2009), pp. 1–6.

[145] Morino, L. Helmholtz decomposition revisited: Vorticity generation and trailing edge condition. *Computational Mechanics 1*, 1 (1986), 65–90.

[146] Mucke, E. P. Geomdir – http://www.geom.uiuc.edu/software/cglist/GeomDir/.

[147] Nese, J. M. Quantifying local predicatability in phase space. *Physica D 35* (1989), 237–250.

[148] Nguyen, D. Q., Fedkiw, R., and Jensen, H. W. Physically based modeling and animation of fire. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 21*, 3 (2002), 721–728.

[149] Nielson, G. M., and Jung, I.-H. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 360–372.

[150] Norton, S. J. Tomographic reconstruction of 2-D vector fields: Applications to flow imaging. *Geophysics Journal International 97*, 1 (1988), 161–168.

[151] Norton, S. J. Unique tomographic reconstruction of vector fields using boundary data. *IEEE Transactions on Image Processing 1*, 3 (1992), 406–412.

[152] Oberkampf, W. L., and Roy, C. J. *Verification and Validation in Scientific Computing*. Cambridge University Press, 2010.

[153] Otto, M., Germer, T., Hege, H.-C., and Theisel, H. Uncertain 2D vector field topology. *Computer Graphics Forum 29*, 2 (2010), 347–356.

[154] Otto, M., Germer, T., and Theisel, H. Uncertain topology of 3D vector fields. In *Proceedings of IEEE Pacific Visualization Symposium* (2011), pp. 67–74.

[155] Palit, B., Basu, A., and Mandal, M. K. Applications of the discrete Hodge Helmholtz decomposition to image and video processing. In *Pattern Recognition and Machine Intelligence*, S. K. Pal, S. Bandyopadhyay, and S. Biswas, Eds., vol. 3776 of *Lecture Notes in Computer Science*. Springer Berlin, 2005, pp. 497–502.

[156] Pang, A. T., Wittenbrink, C. M., and Lodha, S. K. Approaches to uncertainty visualization. *The Visual Computer 13*, 8 (1996), 370–390.

[157] Panton, R. L. *Incompressible flows*. Wiley & Sons, 2005.

[158] Pascucci, V., and Frank, R. J. Global static indexing for real-time exploration of very large regular grids. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing* (New York, NY, USA, 2001), SC '01, ACM, pp. 2–2.

[159] Pascucci, V., and Frank, R. J. Hierarchical indexing for out-of-core access to multi-resolution data. In *Hierarchical and Geometrical Methods in Scientific Visualization*, G. Farin, B. Hamann, and H. Hagen, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2003, pp. 225–241.

[160] Peikert, R., Pobitzer, A., Sadlo, F., and Schindler, B. A comparison of finite-time and finite-size lyapunov exponents. In *Topological Methods in Data Analysis and Visualization III – Theory, Algorithms, and Applications*, P.-T. Bremer, I. Hotz, V. Pascucci, and R. Peikert, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2014, pp. 187–200.

[161] Perry, A. E., and Chong, M. S. Topology of flow patterns in vortex motions and turbulence. *Applied Scientific Research 53*, 3–4 (1994), 357–374.

[162] Perry, A. E., and Tan, D. K. M. Simple three-dimensional vortex motion in coflowing jets and wakes. *Journal of Fluid Mechanics 141* (1984), 197–231.

[163] Petronetto, F., Paiva, A., Lage, M., Tavares, G., Lopes, H., and Lewiner, T. Meshless Helmholtz-Hodge decomposition. *IEEE Transactions on Visualization and Computer Graphics 16*, 2 (2010), 338–342.

[164] Pobitzer, A., Peikert, R., Fuchs, R., Schindler, B., Kuhn, A., Theisel, H., Matkovic, K., and Hauser, H. On the way towards topology-based visualization of unsteady flow - the state of the art. In *Eurographics 2010 State-of-the-Art Reports (STARs)* (2010), pp. 137–154.

[165] Polthier, K., and Preuss, E. Variational approach to vector field decomposition. In *Data Visualization 2000*, W. C. de Leeuw and R. van Liere, Eds., Eurographics. Springer Vienna, 2000, pp. 147–155.

[166] Polthier, K., and Preuss, E. Identifying vector field singularities using a discrete Hodge decomposition. In *Visualization and Mathematics III*, H.-C. Hege and K. Polthier, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2003, pp. 112–134.

[167] Popinet, S. Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries. *Journal of Computational Physics 190*, 2 (2003), 572–600.

[168] Post, F. H., Vrolijk, B., Hauser, H., Laramee, R. S., and Doleisch, H. Feature extraction and visualization of flow fields. In *Eurographics 2002 State-of-the-Art Reports* (2002), pp. 69–100.

[169] Potter, K. *The Visualization of Uncertainty.* PhD thesis, University of Utah, 2010.

[170] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[171] Prince, J. L. Convolution backprojection formulas for 3-D vector tomography with application to MRI. *IEEE Transactions on Image Processing 5*, 10 (1996), 1462–1472.

[172] Ray, N., Vallet, B., Li, W.-C., and Lévy, B. N-symmetry direction field design. *ACM Transactions on Graphics 27*, 2 (2008).

[173] Reininghaus, J., and Hotz, I. Combinatorial 2D vector field topology extraction and simplification. In *Topological Methods in Data Analysis and Visualization – Theory, Algorithms, and Applications*, V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2011, pp. 103–114.

[174] Reininghaus, J., Löwen, C., and Hotz, I. Fast combinatorial vector field topology. *IEEE Transactions on Visualization and Computer Graphics 17*, 10 (2011), 1433–1443.

[175] Sadlo, F., and Peikert, R. Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 1456–1463.

[176] Sadlo, F., and Peikert, R. Visualizing Lagrangian coherent structures and comparison to vector field topology. In *Topology-Based Methods in Visualization II*, H.-C. Hege, K. Polthier, and G. Scheuermann, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 15–30.

[177] Sahner, J., Weinkauf, T., and Hege, H.-C. Galilean invariant extraction and iconic representation of vortex core lines. In *Proceedings of Eurographics / IEEE VGTC Symposium on Visualization (EuroVis)* (2005), pp. 151–160.

[178] Scharstein, R. W. Helmholtz decomposition of surface electric current in electromagnetic scattering problems. In *Proceedings of 23rd Southeastern Symposium on System Theory* (1991), pp. 424–426.

[179] Scheuermann, G., Bobach, T., Hagen, H., Mahrous, K., Hamann, B., Joy, K. I., and Kollmann, W. A tetrahedra-based stream surface algorithm. In *Proceedings of IEEE Visualization* (2001), pp. 151–158.

[180] Scheuermann, G., Hagen, H., and Krüger, H. An interesting class of polynomial vector fields. In *Mathematical Methods for Curves and Surfaces II*. Vanderbilt University Press, 1998, pp. 429–436.

[181] Scheuermann, G., Hagen, H., Krüger, H., Menzel, M., and Rockwood, A. P. Visualization of higher order singularities in vector fields. In *Proceedings of IEEE Visualization* (1997), pp. 67–74.

[182] Scheuermann, G., Krüger, H., Menzel, M., and Rockwood, A. P. Visualizing nonlinear vector field topology. *IEEE Transactions on Visualization and Computer Graphics 4*, 2 (1998), 109–116.

[183] Scheuermann, G., and Tricoche, X. Topological methods for flow visualization. In *The Visualization Handbook*, C. D. Hansen and C. R. Johnson, Eds. Elsevier, 2005, pp. 341–356.

[184] Schönert, M., Besche, H. U., Breuer, T., Celler, F., Eick, B., Felsch, V., Hulpke, A., Mnich, J., Nickel, W., Pfeiffer, G., Polis, U., and Theissen, H. *GAP – Groups, Algorithms, and Programming*, 5th ed. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, Dec 1995.

[185] Schroeder, W., Martin, K., and Lorensen, B. *The Visualization Toolkit*, 4th ed. Kitware, Inc., 2006.

[186] Schwarz, G. *Hodge Decomposition - A Method for Solving Boundary Value Problems*. Springer Berlin, 1995.

[187] Shadden, S. C., Lekien, F., and Marsden, J. E. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D 212* (2005), 271–304.

[188] Söderström, A., Karlsson, M., and Museth, K. A PML-based nonrefelective boundary for free surface fluid animation. *ACM Transactions on Graphics 29*, 5 (2010), 136:1–136:17.

[189] Solomon, J. M., and Szymczak, W. G. Finite difference solutions for the incompressible Navier-Stokes equations using Galerkin techniques. In *5th IMACS International Symposium on Computer Methods for Partial Differential Equations* (June 19-21 1984), Lehigh University.

[190] Sprössig, W. On Helmholtz decompositions and their generalizations - An overview. *Mathematical Methods in the Applied Sciences 33*, 4 (2010), 374–383.

[191] Stam, J. Stable fluids. In *SIGGRAPH 99 Conference Proceedings, Annual Conference Series* (1999), pp. 121–128.

[192] Stam, J. A simple fluid solver based on the FFT. *Journal of Graphics Tools 6*, 2 (2002), 43–52.

[193] Stam, J., and Fiume, E. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH 95 Conference Proceedings, Annual Conference Series* (1995), pp. 129–136.

[194] Stein, R. F., and Nordlund, Å. Realistic solar convection simulations. *Solar Physics 192* (2000), 91–108.

[195] Stephens, A. B., Bell, J. B., Solomon, J. M., and Hackerman, L. B. A finite difference galerkin formulation for the incompressible Navier-Stokes equations. *Journal of Computational Physics 53*, 1 (1984), 152–172.

[196] Stewart, A. M. Longitudinal and transverse components of a vector field. *Sri Lankan Journal of Physics 12* (2011), 33–42.

[197] Szymczak, A. Stable Morse decompositions for piecewise constant vector fields on surfaces. *Computer Graphics Forum 30*, 3 (2011), 851–860.

[198] Szymczak, A., and Zhang, E. Robust Morse decompositions of piecewise constant vector fields. *IEEE Transactions on Visualization and Computer Graphics 18*, 6 (2011), 938–951.

[199] Theisel, H. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum 21*, 3 (2002), 595–604.

[200] Theisel, H., Rössl, C., and Seidel, H.-P. Compression of 2D vector fields under guaranteed topology preservation. *Computer Graphics Forum 22*, 3 (2003), 333–342.

[201] Theisel, H., Rössl, C., and Seidel, H.-P. Using feature flow fields for topological comparison of vector fields. In *Proceedings of the Vision, Modeling, and Visualization Conference* (2003), pp. 521–528.

[202] Theisel, H., Weinkauf, T., Hege, H.-C., and Seidel, H.-P. Saddle connectors – an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proceedings of IEEE Visualization* (2003), pp. 225–232.

[203] Theisel, H., Weinkauf, T., Hege, H.-C., and Seidel, H.-P. Stream line and path line oriented topology for 2D time-dependent vector fields. In *Proceedings of IEEE Visualization* (2004), pp. 321–328.

[204] Tong, Y., Lombeyda, S., Hirani, A. N., and Desbrun, M. Discrete multiscale vector field decomposition. *ACM Transactions on Graphics 22*, 3 (2003), 445–452.

[205] Tricoche, X., Scheuermann, G., and Hagen, H. Higher order singularities in piecewise linear vector fields. In *The Mathematics of Surfaces IX*, R. Cipolla and R. Martin, Eds. Springer London, 2000, pp. 99–113.

[206] Tricoche, X., Scheuermann, G., and Hagen, H. A topology simplification method for 2D vector fields. In *Proceedings of IEEE Visualization* (2000), pp. 359–366.

[207] VAN KAN, J. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal on Scientific and Statistical Computing 7*, 3 (1986), 870–891.

[208] VAN WIJK, J. J. Image based flow visualization. *ACM Transactions on Graphics 21*, 3 (2002), 745–754.

[209] VERMA, V., AND PANG, A. T. Comparative flow visualization. *IEEE Transactions on Visualization and Computer Graphics 10*, 6 (2004), 609–624.

[210] VON HELMHOLTZ, H. Über Integrale der hydrodynamischen Gleichungen, welche den Wirbelbewegungen entsprechen. *Journal für die reine und angewandte Mathematik 1858*, 55 (Jan 1858), 25–55.

[211] VON HELMHOLTZ, H. On integrals of the hydrodynamical equations, which express vortex-motion. *Philosophical Magazine and Journal of Science 33*, 226 (1867), 485–512.

[212] WANG, K., WEIWEI, TONG, Y., DESBRUN, M., AND SCHRÖDER, P. Edge subdivision schemes and the construction of smooth vector fields. *ACM Transactions on Graphics 25*, 3 (2006), 1041–1048.

[213] WEINKAUF, T., THEISEL, H., HEGE, H.-C., AND SEIDEL, H.-P. Boundary switch connectors for topological visualization of complex 3D vector fields. In *Proceedings of Eurographics / IEEE TCVG Symposium on Visualization (VisSym)* (2004), pp. 183–192.

[214] WEINKAUF, T., THEISEL, H., HEGE, H.-C., AND SEIDEL, H.-P. Topological construction and visualization of higher order 3D vector fields. *Computer Graphics Forum 23*, 3 (2004), 469–478.

[215] WEINKAUF, T., THEISEL, H., SHI, K., HEGE, H.-C., AND SEIDEL, H.-P. Extracting higher order critical points and topological simplification of 3D vector fields. In *Proceedings of IEEE Visualization* (2005), pp. 559–566.

[216] WEYL, H. The method of orthogonal projection in potential theory. *Duke Mathematical Journal 7*, 7 (1940), 411–444.

[217] WIEBEL, A. Feature detection in vector fields using the Helmholtz-Hodge decomposition. Diplomarbeit, University of Kaiserslautern, 2004.

[218] WIEBEL, A., CHAN, R., WOLF, C., ROBITZKI, A., STEVENS, A., AND SCHEUERMANN, G. Topological flow structures in a mathematical model for rotation-mediated cell aggregation. In *Topological Methods in Data Analysis and Visualization – Theory, Algorithms, and Applications*, V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2011, pp. 193–204.

[219] WIEBEL, A., GARTH, C., AND SCHEUERMANN, G. Localized flow analysis of 2D and 3D vector fields. In *Proceedings of Eurographics / IEEE VGTC Symposium on Visualization (EuroVis)* (2005), pp. 143–150.

[220] WIEBEL, A., GARTH, C., AND SCHEUERMANN, G. Computation of localized flow for steady and unsteady vector fields and its applications. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 641–651.

[221] Wittenbrink, C. M., Pang, A. T., and Lodha, S. K. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics 2*, 3 (1996), 266–279.

[222] Yoo, C. S., Richardson, E. S., Sankaran, R., and Chen, J. H. A DNS study on the stabilization mechanism of a turbulent lifted ethylene jet flame in highly-heated coflow. *Proceedings of the Combustion Institute 33*, 1 (2011), 1619–1627.

[223] Yoo, C. S., Sankaran, R., and Chen, J. H. Three-dimensional direct numerical simulation of a turbulent lifted hydrogen jet flame in heated coflow: Flame stabilization and structure. *Journal of Fluid Mechanics 640* (2009), 453–481.

[224] Yu, H., Wang, C., Grout, R. W., Chen, J. H., and Ma, K.-L. In situ visualization for large-scale combustion simulations. *IEEE Computer Graphics and Applications 30*, 3 (2010), 45–57.

[225] Zhang, E., Mischaikow, K., and Turk, G. Vector field design on surfaces. *ACM Transactions on Graphics 25*, 4 (2006), 1294–1326.

[226] Zuk, T., Downton, J., Gray, D., Carpendale, S., and Liang, J. D. Exploration of uncertainty in bidirectional vector fields. In *Proceedings of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (2008), vol. 6809.